9. 기본키 취득

MyBatis 는 OracleDB 에서 selectKey 속성을 이용하여 INSERT 할 때 데이터베이스에서 선택된 기본키를 취득할 수 있다.

(1) 데이터베이스 테이블

① 회원 테이블과 권한 테이블

```
CREATE TABLE
    mybatismember(user_no
    NUMBER,
    user_id VARCHAR2(50) NOT NULL,
    user_pw VARCHAR2(50) NOT NULL,
    user_name VARCHAR2(100) NOT NULL,
    coin NUMBER(10) DEFAULT 0,
    reg_date DATE DEFAULT SYSDATE,
    upd_date DATE DEFAULT SYSDATE,
    enabled CHAR(1) DEFAULT '1',
    PRIMARY KEY (user_no)
);

CREATE TABLE
    mybatismember_auth(user_no
    NUMBER NOT NULL, auth
    VARCHAR2(50) NOT NULL
);

ALTER TABLE mybatismember_auth ADD CONSTRAINT fk_mybatismember_auth_user_no
FOREIGN KEY(user_no) REFERENCES mybatismember(user_no);

create sequence mybatismember_seq
start with 1
increment by 1;
```

(2) MyBatis 설정

/resource/mybatis-config.xml

```
<!-- mapUnderscoreToCamelCase 프로퍼티 값을 true 로 지정 -->
<settings>
	<setting name="mapUnderscoreToCamelCase" value="true" />
</settings>
<!-- TypeAlias 로 맵핑 파일에서 반복적으로 사용될 패키지의 이름을 정의한다. -->
<typeAliases>
	<package name="com.zeus.domain" />
</typeAliases>
```

(3) MyBatis 구현

① Mapper 인터페이스 메서드

/src/main/java/com/zeus/mapper/**MemberMapper.java**

```
public interface MemberMapper {
	public void create(Member member) throws Exception;
	public void createAuth(MemberAuth memberAuth) throws Exception;
}
```

· 매핑 파일

src/main/**resources/com/zeus/mapper/MemberMapper.xml**

```
<!-- 데이터베이스에서 기본키를 인수로 전달한 자바빈의 프로퍼티를 통해 취득한다. -->
<insert id="create" parameterType="Member">
	<selectKey keyProperty="userNo" resultType="int" order="BEFORE">
	    select mybatismember_seq.NEXTVAL FROM DUAL
	</selectKey>
	INSERT INTO mybatismember
		(
		  user_no,
		  user_id,
		  user_pw,
		  user_name
		)
	VALUES
		(
```

```
                    #{userNo},
                    #{userId},
                    #{userPw},
                    #{userName}
                )
    </insert>
    <insert id="createAuth">
        INSERT INTO mybatismember_auth
            (
                user_no,
                auth
            )
        VALUES
                (
                #{userNo},
                #{auth}
                )
    </insert>
```

(4) 회원 등록 구현 설명

① 등록 화면

· 요청

```
http://localhost:8080/user/register
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/**MemberController.java**

```
    @RequestMapping(value = "/register", method = RequestMethod.GET)
    public void registerForm(Member member, Model model) throws Exception {
        log.info("UserRegisterForm");
    }
```

② 등록

· 요청(POST)

```
http://localhost:8080/user/register
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/**MemberController.java**

```
	@RequestMapping(value = "/register", method = RequestMethod.POST)
	public String register(Member member, Model model) throws Exception {
		service.register(member);

		model.addAttribute("msg", "등록이 완료되었습니다.");

		return "user/success";
	}
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service/**MemberService.java**

```
public interface MemberService {
	public void register(Member member) throws Exception;
}
```

· 서비스 구현 메서드

/src/main/java/com/zeus/service/**MemberServiceImpl.java**

```
	@Transactional
	@Override
	public void register(Member member) throws Exception {
		mapper.create(member);

		MemberAuth memberAuth = new MemberAuth();

		memberAuth.setUserNo(member.getUserNo());
		memberAuth.setAuth("ROLE_USER");

		mapper.createAuth(memberAuth);
	}
```

(5) 구현 소스

/resource

mybatis-config.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>

	<!-- mapUnderscoreToCamelCase 프로퍼티 값을 true로 지정 -->
	<settings>
		<setting name="mapUnderscoreToCamelCase" value="true" />
	</settings>

	<!-- TypeAlias로 맵핑 파일에서 반복적으로 사용될 패키지의 이름을 정의한다. -->
	<typeAliases>
		<package name="com.zeus.domain" />
	</typeAliases>

</configuration>
```

/src/main/java/com/zeus/domain

Member.java

```
package com.zeus.domain;

import java.util.Date;
import java.util.List;

import lombok.Getter;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@ToString
public class Member {
	private int userNo;
	private String userId;
	private String userPw;
	private String userName;
	private Date regDate;
	private Date updDate;

	private List<MemberAuth> authList;
}
```

MemberAuth.java

```
package com.zeus.domain;

import lombok.Getter;
```

```java
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@ToString
public class MemberAuth {

	private int userNo;
	private String auth;

}
```

/src/main/java/com/zeus/controller

MemberController.java

```java
package com.zeus.controller;

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 com.zeus.domain.Member;
import com.zeus.service.MemberService;

import lombok.extern.java.Log;

@Log
@Controller
@RequestMapping("/user")
public class MemberController {

	@Autowired
	private MemberService service;

	@RequestMapping(value = "/register", method = RequestMethod.GET)
	public void registerForm(Member member, Model model) throws Exception {
		log.info("UserRegisterForm");
	}

	@RequestMapping(value = "/register", method = RequestMethod.POST)
	public String register(Member member, Model model) throws Exception {
		service.register(member);

		model.addAttribute("msg", "등록이 완료되었습니다.");

		return "user/success";
	}

}
```

/src/main/java/com/zeus/service

MemberService.java

```
package com.zeus.service;

import com.zeus.domain.Member;

public interface MemberService {
	public void register(Member member) throws Exception;
}
```

MemberServiceImpl.java

```
package com.zeus.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.zeus.domain.Member;
import com.zeus.domain.MemberAuth;
import com.zeus.mapper.MemberMapper;

@Service
public class MemberServiceImpl implements MemberService {

	@Autowired
	private MemberMapper mapper;

	@Transactional
	@Override
	public void register(Member member) throws Exception
		{mapper.create(member);

		MemberAuth memberAuth = new MemberAuth();

		memberAuth.setUserNo(member.getUserNo());
		memberAuth.setAuth("ROLE_USER");

		mapper.createAuth(memberAuth);
	}
}
```

/src/main/java/com/zeus/mapper

MemberMapper.java

```
package com.zeus.mapper;

import com.zeus.domain.Member;
import com.zeus.domain.MemberAuth;

public interface MemberMapper {
	public void create(Member member) throws Exception;
	public void createAuth(MemberAuth memberAuth) throws Exception;
}
```

src/main/resources/com/zeus/mapper

MemberMapper.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="com.zeus.mapper.MemberMapper">

	<!-- 데이터베이스에서 기본키를 인수로 전달한 자바빈의 프로퍼티를 통해 취득한다. -->
	<!-- <insert id="create" useGeneratedKeys="true" keyProperty="userNo"> -->
	<insert id="create" parameterType="Member">
		<selectKey keyProperty="userNo" resultType="int" order="BEFORE">
		select mybatismember_seq.NEXTVAL FROM DUAL
		</selectKey>
		INSERT INTO mybatismember
			(
			  user_no,
			  user_id,
			  user_pw,
			  user_name
			)
		VALUES
			(
			  #{userNo},
			  #{userId},
			  #{userPw},
			  #{userName}
			)
	</insert>

	<insert id="createAuth">
		INSERT INTO mybatismember_auth
		    (
		      user_no,
		      auth
		    )
		VALUES
		    (
		      #{userNo},
		      #{auth}
		    )
	</insert>
</mapper>
```

/src/main/webapp/WEB-INF/views/user

register.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Mybatis USER 등록</title>
```

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
	$(document).ready(function() {

		var formObj = $("#member");

		$("#btnRegister").on("click", function()
			{ formObj.attr("action",
			"/user/register");formObj.attr("method",
			"post"); formObj.submit();
		});

	});
</script>
</head>
<body>
	<h3>회원 등록</h3>

	<form:form modelAttribute="member" action="register">
		<table>
			<tr>
				<td>userid</td>
				<td><form:input path="userId" /></td>
				<td><font color="red"><form:errors path="userId"
/></font></td>
			</tr>
			<tr>
				<td>userpw</td>
				<td><form:input path="userPw" /></td>
				<td><font color="red"><form:errors path="userPw"
/></font></td>
			</tr>
			<tr>
				<td>username</td>
				<td><form:input path="userName" /></td>
				<td><font color="red"><form:errors path="userName"
/></font></td>
			</tr>
		</table>
	</form:form>

	<div>
		<button type="submit" id="btnRegister">등록</button>
	</div>
</body>
</html>
```

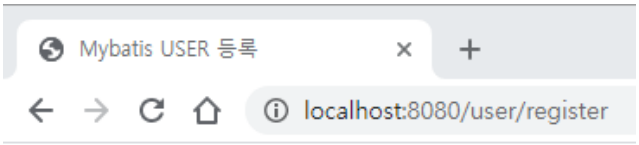
success.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Mybatis USER 등록</title>
```

```
</head>
<body>
        <h3>${msg}</h3>
</body>
</html>
```

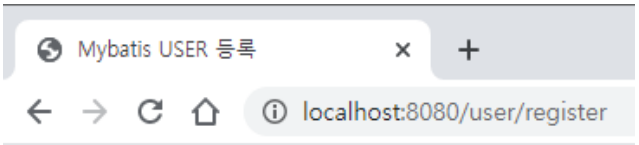
실행 결과

http://localhost:8080/user/register



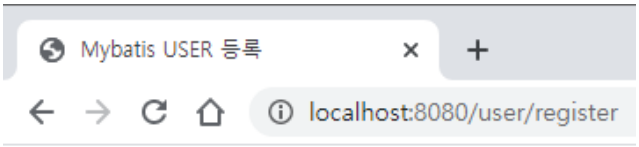
회원 등록

userid
userpw
username
등록



회원 등록

userid jws
userpw 1234
username 제우스
등록



등록이 완료되었습니다.

10. sql 요소

이 요소는 다른 구문에 포함될 수 있는 재사용 가능한 SQL 코드 조각을 정의하는데 사용될 수 있다. 로드 단계에서 정적으로 매개변수화 할 수 있다. 다른 프로퍼티 값을 포함한 인스턴스에 따라 달라질 수 있다. 예를 들면

```
<sql id="userColumns">${alias}.id, ${alias}.username, ${alias}.password</sql>
```

SQL 구문을 다른 구문에 포함시킬 수 있다. 예를 들면

```
<select id="selectUser" resultType="map">
    SELECT
      <include refid="userColumns"><property name="alias" value="t1"/><include>
      <include refid="userColumns"><property name="alias" value="t2"/><include>
    FROM some_table  t1
      cross join some_table t2
</select>
```

프로퍼티 값은 include refid 속성 또는 include 절 내의 프로퍼티 값에 사용될 수도 있다. 예를 들면

```
<sql id="sometable">
    ${prefix}Table
</sql>

<sql id="someinclude">
    FROM
      <include refid="${include_target}"/>
</sql>

<select id="select" resultType="map">
    SELECT
        field1, field2, field3
    <include refid="someinclude">
        <property name="prefix" value="some"/>
        <property name="include_target" value="sometable"/>
    </include>
</select>
```

11. Parameters

지금까지의 모든 문에서 간단한 매개변수의 예를 보았다. 매개변수는 MyBatis 에서 매우 강력한 요소이다.

```
<select id="selectUsers" resultType="User">
    SELECT id, username, password
    FROM users
    WHERE id=#{id}
</select>
```

위의 예에서 간단히 명명된 매개변수 매핑을 보여준다. parameterType 은 int 로 설정할 수 있다. Integer 및 String 과 같은 데이터 타입에는 관련 프로퍼티가 없으므로 매개변수의 전체값을 대체한다. 그러나 복잡한 객체를 전달하면 동작이 다르다. 예를 들면

```
<insert id="insertUser" parameterType="User">
    INSERT INTO users(id, username, password)
    VALUES (#{id}, #{username}, #{password})
</insert>
```

User 타입의 매개변수 객체가 해당 구문으로 전달된 경우 id, username, password 프로퍼티가 조회되고 해당 값이 PreparedStatement 매개변수로 전달된다.

구문에 매개변수를 전달하는 것은 간단하다. 그러나 매개변수 매핑에는 다른 기능이 있다.

첫째, MyBatis 의 다른 부분과 마찬가지로 매개변수는 보다 구체적인 데이터 타입을 지정할 수 있다.

```
#{property.javaType=int.jdbcType=NUMERIC}
```

(1) 문자열 대체

기본적으로 **#{ }**구문을 사용하면 MyBatis 가 PreparedStatement 프로퍼티를 생성하고 PreparedStatement 매개변수 (?)에 대해 값을 설정한다. 이것이 안정하고 빨라 선호되지만 때로는 수정되지 않은 문자열을 SQL 문에 직접 주입하려고 할 수도 있다. 예를 들어, ORDER BY 의 경우 다음과 같은 것을 사용할 수도 있다.

```
ORDER BY ${columnName}
```

여기서 MyBatis 는 문자열을 수정하거나 이스케이프 처리하지 않는다. 문자열 대체는 SQL 문의 메타데이터(테이블명 또는 컬럼명)가 동적일 때 유용하다.

```
@Select("select * from user where id=#{id}")
User findById(@Param("id") long id);

@Select("select * from user where name=#{name}")
User findByName(@Param("name") String name);
```

```
@Select("select * from user where email=#{email}")
User findByEmail(@Param("email") String email);
```

다음과 같이 정리할 수 있다.

```
@Select("select * from user where ${column}=#{value}")
User findByColumn(@Param("column") String column, @Param("value") String value);
```

${column}이 직접 대체되고 #{value}가 "prepared(처리가 준비됨)"된다. 따라서 다음과 같은 방법으로 작업을 수행할 수 있다.

```
User userOfId1 = userMapper.findByColumn("id", 1L);
User userOfNameKid = userMapper.findByColumn("name", "kid");
User userOfEmail = userMapper.findByColumn("email", "jeus@xxxx.com")
```

※ 주의
이 방법으로 사용자의 입력을 받아 수정되지 않은 구문에 제공하는 것은 안전하지 않다. 이로 인해 잠재적인 SQL 인젝션 공격이 발생하므로 이 필드에서 반드시 사용자 입력을 허용하지 않거나 항상 이스케이프(특수문자)를 검사해야 한다.

12. ResultMaps

resultMap 요소는 MyBatis 에서 가장 중요하고 강력한 요소이다. JDBC 가 ResultSet 에서 데이터를 검색하는 데 필요한 코드의 90%를 제거할 수 있으며, 경우에 따라 JDBC 가 지원하지 않는 작업도 수행할 수 있다. 실제로 복잡한 구문의 조인 매핑과 같은 코드를 작성하려면 수천 줄의 코드가 포함될 수 있다. ResultMaps 의 설계는 간단한 구문에서 명시적인 결과 매핑이 필요하지 않으며 복잡한 구문은 관계를 설명하는 데 필요하다.

명시적인 resultMap 이 없는 간단히 매핑된 구문의 예를 들면 다음과 같다.

```
<select id="selectUsers" resultType="map">
    SELECT id, username, hashedPassword
    FROM some_table
    WHERE id=#{id}
</select>
```

이러한 구문은 resultType 속성으로 지정된 대로 모든 컬럼이 HashMap 의 키에 자동으로 매핑된다. 많은 경우에 유용하지만 HashMap 은 매우 좋은 도메인 모델을 만들지 낳는다. 애플리케이션이 도메인 모델에 JavaBeans 또는 POJO(Plain Old Java Objects)를 사용할 가능성이 높다. MyBatis 는 두 가지를 모드 지원한다.

JavaBean 의 경우

```
public class User {
        private int id;
        private String username;
        private String hashedPassword;
        public int getId() {
                return id;
        }
        public void setId(int id) {
                this.id = id;
        }
        public String getUsername()
                {return username;
        }
        public void setUsername(String username)
                {this.username = username;
        }
        public String getHashedPassword()
                {return hashedPassword;
        }
```

```
        public void setHashedPassword(String hashedPassword)
                {this.hashedPassword = hashedPassword;
        }
}
```

JavaBean 인 클래스에느 id, username, hashedPassword 의 3 개의 프로퍼티가 있다. 이들은 select 문의 컬럼명과 정확히 일치한다.

이러한 JavaBean 은 HashMap 처럼 쉽게 ResultSet 에 매핑될 수 있다.

```
<select id="selectUsers" resultType="com.model.User">
    SELECT id, username, hashedPassword
    FROM user
    WHERE id=#{id}
</select>
```

그리고 TypeAliases 기능을 사용하면 클래스 전체 경로를 계속 입력할 필요가 없다.

```
<!-- In Config XML file -->
<typeAlias type="com.model.User" alias="User"/>

<!-- In SQL Mapping XML file -->
<select id="selectUsers" resultType="com.model.User">
    SELECT id, username, hashedPassword
    FROM user
    WHERE id=#{id}
</select>
```

이 경우 MyBatis 는 자동으로 이름 기준으로 JavaBean 프로퍼티에 컬럼을 자동 매핑하기 위해 백그라운드에서 ResultMap 을 자동으로 작성한다. 컬럼명이 정확하게 일치하지 않으면 컬럼명에 select 절 별칭(표준 SQL 기능)을 사용하여 레이블을 일치시킬 수 있다.

```
<select id="selectUsers" resultType="User">
    SELECT
        user_id AS  "id",
        user_name AS "username",
        hashed_password AS "hashedPassword"
    FROM user
    WHERE id=#{id}
</select>
```

컬럼명이 불일치를 해결하는 다른 방법은 외부 resultMap 요소를 활용하는 것이다.

```
<resultMap id="userResultMap" type="User">
    <result property="id" column="user_id">
    <result property="username" column="user_name">
    <result property="password" column="hashed_password">
</resultMap>
```

그리고 이를 참조하는 구문은 resultMap 속성을 사용하여 이를 수행한다.

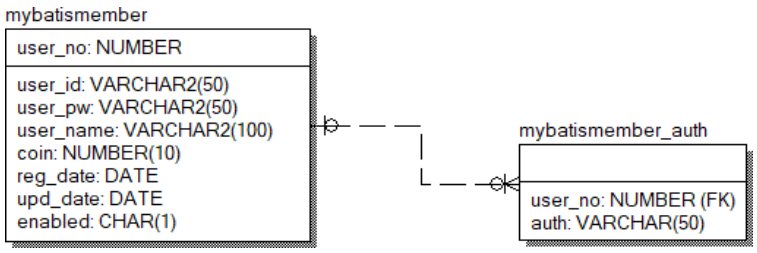
```
<select id="selectUsers" resultMap="userResultMap">
    SELECT user_id, user_name, hased_password
    FROM user
    WHERE id=#{id}
</select>
```

13. 일대다 관계 테이블 매핑

MyBatis 기능을 활용하여 매핑 파일을 적절하게 정의하면 일대다 관계 테이블 매핑을 쉽게 처리할 수 있다.

(1) 데이터베이스 테이블

① 회원 테이블과 권한 테이블



(2) MyBatis 설정

/resource/

mybatis-config.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>

	<!-- mapUnderscoreToCamelCase 프로퍼티 값을 true로 지정 -->
	<settings>
		<setting name="mapUnderscoreToCamelCase" value="true" />
	</settings>

	<!-- TypeAlias로 맵핑 파일에서 반복적으로 사용될 패키지의 이름을 정의한다. -->
	<typeAliases>
		<package name="com.zeus.domain" />
	</typeAliases>

</configuration>
```

(3) MyBatis 구현

① Mapper 인터페이스

/src/main/java/com/zeus/mapper/MemeberMapper.java

```
public Member read(int userNo) throws Exception;
public void update(Member member) throws Exception;
public void delete(int userNo) throws Exception;
public List<Member> list() throws Exception;
public void deleteAuth(int userNo) throws Exception;
```

② 매핑 파일

src/main/resources/com/zeus/mapper/MemberMapper.xml

```
<!-- 외부 조인을 사용하여 얻은 검색 결과에 대한 매핑을 정의한다. -->
<resultMap type="Member" id="memberMap">
        <id property="userNo" column="user_no" />
        <result property="userNo" column="user_no" />
        <result property="userId" column="user_id" />
        <result property="userPw" column="user_pw" />
        <result property="userName" column="user_name" />
        <result property="regDate" column="reg_date" />
        <result property="updDate" column="upd_date" />
        <collection property="authList" resultMap="authMap">
        </collection>
</resultMap>

<resultMap type="MemberAuth" id="authMap">
        <result property="userNo" column="user_no" />
        <result property="auth" column="auth" />
</resultMap>
```

(4) 회원

① 회원 목록 화면

· 요청

```
http://localhost:8080/user/list
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/MemberController.java

```
	@RequestMapping(value = "/list", method = RequestMethod.GET)
	public void list(Model model) throws Exception {
		model.addAttribute("list", service.list());
	}
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service/MemberService.java

```
public List<Member> list() throws Exception;
```

· 서비스 구현 메서드

/src/main/java/com/zeus/service/MemberServiceImpl.java

```
	@Override
	public List<Member> list() throws Exception
		{return mapper.list();
	}
```

· Mapper 인터페이스 메서드

/src/main/java/com/zeus/mapper/MemberMapper.java

```
	public List<Member> list() throws Exception;
```

· 매핑 파일

src/main/resources/com/zeus/mapper/MemberMapper.xml

```
	<select id="list" resultType="Member">
		SELECT user_no,
```

```
                    user_id,
                    user_pw,
                    user_name,
                    reg_date
                FROM mybatismember
            ORDER BY reg_date DESC
        </select>
```

② 회원 정보 상세 화면

· 요청

```
http://localhost:8080/user/read
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/MemberController.java

```
        @RequestMapping(value = "/read", method = RequestMethod.GET)
        public void read(int userNo, Model model) throws Exception {
                model.addAttribute(service.read(userNo));
        }
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service/MemberService.java

```
        public Member read(int userNo) throws Exception;
```

· 서비스 구현 메서드

/src/main/java/com/zeus/service/MemberServiceImpl.java

```
        @Override
        public Member read(int userNo) throws Exception
                {return mapper.read(userNo);
        }
```

· Mapper 인터페이스 메서드

/src/main/java/com/zeus/mapper/MemberMapper.java

```
	public Member read(int userNo) throws Exception;
```

· 매핑 파일

src/main/resources/com/zeus/mapper/MemberMapper.xml

```
	<!-- 외부 조인을 사용하여 얻은 검색 결과에 대한 매핑을 정의한다. -->
	<resultMap type="Member" id="memberMap">
		<id property="userNo" column="user_no" />
		<result property="userNo" column="user_no" />
		<result property="userId" column="user_id" />
		<result property="userPw" column="user_pw" />
		<result property="userName" column="user_name" />
		<result property="regDate" column="reg_date" />
		<result property="updDate" column="upd_date" />
		<collection property="authList" resultMap="authMap">
		</collection>
	</resultMap>
	<resultMap type="MemberAuth" id="authMap">
		<result property="userNo" column="user_no" />
		<result property="auth" column="auth" />
	</resultMap>

	<!-- 부모 객체와 관계 객체를 작성하는 데 필요한 모든 레코드를 검색하도록 외부 조인을 사용한 SQL 을 정의한다. -->
	<select id="read" resultMap="memberMap">
		SELECT mem.user_no,
			mem.user_id,
			user_pw,
			user_name,
			reg_date,
			upd_date,
			auth
		  FROM mybatismember mem LEFT OUTER JOIN member_auth auth ON mem.user_no = auth.user_no
		 WHERE mem.user_no = #{userNo}
	</select>
```

③ 회원 정보 수정 화면

· 요청

```
http://localhost:8080/user/modify
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/MemberController.java

```
@RequestMapping(value = "/modify", method = RequestMethod.GET)
public void modifyForm(int userNo, Model model) throws Exception {
        model.addAttribute(service.read(userNo));
}
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service/MemberService.java

```
public Member read(int userNo) throws Exception;
```

· 서비스 구현 메서드

/src/main/java/com/zeus/service/MemberServiceImpl.java

```
@Override
public Member read(int userNo) throws Exception
        {return mapper.read(userNo);
}
```

· Mapper 인터페이스 메서드

/src/main/java/com/zeus/mapper/MemberMapper.java

```
public Member read(int userNo) throws Exception;
```

· 매핑 파일

src/main/resources/com/zeus/mapper/MemberMapper.xml

```
회원 정보 상세 화면과 동일
```

④ 회원 정보 수정

· 요청(POST)

```
http://localhost:8080/user/modify
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/MemberController.java

```
	@RequestMapping(value = "/modify", method = RequestMethod.POST)
	public String modify(Member member, Model model) throws Exception {
		service.modify(member);
		model.addAttribute("msg", "수정이 완료되었습니다.");
		return "user/success";
	}
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service/MemberService.java

```
	public void modify(Member member) throws Exception;
```

· 서비스 구현 메서드

/src/main/java/com/zeus/service/MemberServiceImpl.java

```
	@Transactional
	@Override
	public void modify(Member member) throws Exception {
		mapper.update(member);
		int userNo = member.getUserNo();
		mapper.deleteAuth(userNo);
		List<MemberAuth> authList = member.getAuthList();

		for (int i = 0; i < authList.size(); i++) {
			MemberAuth memberAuth = authList.get(i);
			String auth = memberAuth.getAuth();
			if (auth == null) {
				continue;
			}
			if (auth.trim().length() == 0) {
				continue;
```

```
                    }
                    memberAuth.setUserNo(userNo);
                    mapper.createAuth(memberAuth);
                }
            }
```

· Mapper 인터페이스 메서드

/src/main/java/com/zeus/mapper/MemberMapper.java

```
    public void update(Member member) throws Exception;
    public void deleteAuth(int userNo) throws Exception;
    public void createAuth(MemberAuth memberAuth) throws Exception;
```

· 매핑 파일

src/main/resources/com/zeus/mapper/MemberMapper.xml

```
    <update id="update">
        UPDATE mybatismember
           SET user_name =#{userName}
         WHERE user_no = #{userNo}
    </update>
    <insert id="createAuth">
        INSERT INTO mybatismember_auth
            (
              user_no,
              auth
            )
        VALUES
            (
               #{userNo},
               #{auth}
            )
    </insert>
    <delete id="deleteAuth">
        DELETE FROM mybatismember_auth
         WHERE user_no = #{userNo}
    </delete>
```

⑤ 회원 정보 삭제

· 요청(POST)

```
http://localhost:8080/user/remove
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller/MemberController.java

```
	@RequestMapping(value = "/remove", method = RequestMethod.POST)
	public String remove(int userNo, Model model) throws Exception {
		service.remove(userNo);
		model.addAttribute("msg", "삭제가 완료되었습니다.");
		return "user/success";
	}
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service/MemberService.java

```
	public void remove(int userNo) throws Exception;
```

· 서비스 구현 메서드

/src/main/java/com/zeus/service/MemberServiceImpl.java

```
	@Transactional
	@Override
	public void remove(int userNo) throws Exception
		{ mapper.deleteAuth(userNo); // 삭제 순서
		mapper.delete(userNo);
	}
```

· Mapper 인터페이스 메서드

/src/main/java/com/zeus/mapper/MemberMapper.java

```
	public void delete(int userNo) throws Exception;
	public void deleteAuth(int userNo) throws Exception;
```

· 매핑 파일

src/main/resources/com/zeus/mapper/MemberMapper.xml

```
<delete id="delete">
        DELETE FROM mybatismember
         WHERE user_no = #{userNo}
</delete>

<delete id="deleteAuth">
        DELETE FROM mybatismember_auth
         WHERE user_no = #{userNo}
</delete>
```

(5) 구현 소스(변경 추가 사항)

/src/main/java/com/zeus/controller

MemberController.java

```
package com.zeus.controller;

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 com.zeus.domain.Member;
import com.zeus.service.MemberService;

import lombok.extern.java.Log;

@Log
@Controller
@RequestMapping("/user")
public class MemberController {

	@Autowired
	private MemberService service;

	@RequestMapping(value = "/register", method = RequestMethod.GET)
	public void registerForm(Member member, Model model) throws Exception {
		log.info("UserRegisterForm");
	}

	@RequestMapping(value = "/register", method = RequestMethod.POST)
	public String register(Member member, Model model) throws Exception {
		service.register(member);

		model.addAttribute("msg", "등록이 완료되었습니다.");

		return "user/success";
	}

	@RequestMapping(value = "/list", method = RequestMethod.GET)
	public void list(Model model) throws Exception {
		model.addAttribute("list", service.list());
	}

	@RequestMapping(value = "/read", method = RequestMethod.GET)
	public void read(int userNo, Model model) throws Exception {
		model.addAttribute(service.read(userNo));
	}

	@RequestMapping(value = "/remove", method = RequestMethod.POST)
	public String remove(int userNo, Model model) throws Exception {
		service.remove(userNo);

		model.addAttribute("msg", "삭제가 완료되었습니다.");

		return "user/success";
	}
```

```
	@RequestMapping(value = "/modify", method = RequestMethod.GET)
	public void modifyForm(int userNo, Model model) throws Exception {
		model.addAttribute(service.read(userNo));
	}

	@RequestMapping(value = "/modify", method = RequestMethod.POST)
	public String modify(Member member, Model model) throws Exception {
		service.modify(member);

		model.addAttribute("msg", "수정이 완료되었습니다.");

		return "user/success";
	}

}
```

/src/main/java/com/zeus/service

MemberService.java

```
package com.zeus.service;

import java.util.List;

import com.zeus.domain.Member;

public interface MemberService {
	public void register(Member member) throws Exception;

	public Member read(int userNo) throws Exception;

	public void modify(Member member) throws Exception;

	public void remove(int userNo) throws Exception;

	public List<Member> list() throws Exception;
}
```

MemberServiceImpl.java

```
package com.zeus.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import com.zeus.domain.Member;
import com.zeus.domain.MemberAuth;
import com.zeus.mapper.MemberMapper;

@Service
public class MemberServiceImpl implements MemberService {

	@Autowired
```

```
	private MemberMapper mapper;

	@Transactional
	@Override
	public void register(Member member) throws Exception
		{mapper.create(member);

		MemberAuth memberAuth = new MemberAuth();

		memberAuth.setUserNo(member.getUserNo());
		memberAuth.setAuth("ROLE_USER");

		mapper.createAuth(memberAuth);
	}

	@Override
	public Member read(int userNo) throws Exception
		{return mapper.read(userNo);
	}

	@Transactional
	@Override
	public void modify(Member member) throws Exception
		{mapper.update(member);

		int userNo = member.getUserNo();

		mapper.deleteAuth(userNo);

		List<MemberAuth> authList = member.getAuthList();
		for (int i = 0; i < authList.size(); i++)
			{ MemberAuth memberAuth = authList.get(i);

			String auth = memberAuth.getAuth();

			if (auth == null) {
				continue;
			}
			if (auth.trim().length() == 0)
				{continue;
			}
			memberAuth.setUserNo(userNo);
			mapper.createAuth(memberAuth);
		}
	}

	@Transactional
	@Override
	public void remove(int userNo) throws Exception
		{mapper.deleteAuth(userNo);  // 삭제 순서
		mapper.delete(userNo);
	}

	@Override
```

```
        public List<Member> list() throws Exception
                {return mapper.list();
        }
}
```

/src/main/java/com/zeus/mapper

MemberMapper.java

```
package com.zeus.mapper;

import java.util.List;

import com.zeus.domain.Member;
import com.zeus.domain.MemberAuth;

public interface MemberMapper {
        public void create(Member member) throws Exception;
        public void createAuth(MemberAuth memberAuth) throws Exception;
        public List<Member> list() throws Exception;
        public Member read(int userNo) throws Exception;
        public void update(Member member) throws Exception;
        public void delete(int userNo) throws Exception;
        public void deleteAuth(int userNo) throws Exception;
}
```

src/main/resources/com/zeus/mapper

MemberMapper.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="com.zeus.mapper.MemberMapper">

        <!-- 외부 조인을 사용하여 얻은 검색 결과에 대한 매핑을 정의한다. -->
        <resultMap type="Member" id="memberMap">
                <id property="userNo" column="user_no" />
                <result property="userNo" column="user_no" />
                <result property="userId" column="user_id" />
                <result property="userPw" column="user_pw" />
                <result property="userName" column="user_name" />
                <result property="regDate" column="reg_date" />
                <result property="updDate" column="upd_date" />
                <collection property="authList" resultMap="authMap">
                </collection>
        </resultMap>

        <resultMap type="MemberAuth" id="authMap">
                <result property="userNo" column="user_no" />
                <result property="auth" column="auth" />
        </resultMap>

        <!-- 부모 객체와 관계 객체를 작성하는 데 필요한 모든 레코드를 검색하도록 외부 조인을
```

```
사용한 SQL을 정의한다. -->
	<select id="read" resultMap="memberMap">
		SELECT mem.user_no,
			mem.user_id,
			user_pw,
			user_name,
			reg_date,
			upd_date,
			auth
		 FROM mybatismember mem LEFT OUTER JOIN mybatismember_auth auth ON
mem.user_no = auth.user_no
		 WHERE mem.user_no = #{userNo}
	</select>

	<select id="list" resultType="Member">
	    SELECT user_no,
			user_id,
			user_pw,
			user_name,
			reg_date
		 FROM mybatismember
	  ORDER BY reg_date DESC
	</select>

	<update id="update">
		UPDATE mybatismember
		   SET user_name =#{userName}
		 WHERE user_no = #{userNo}
	</update>

	<delete id="delete">
		DELETE FROM mybatismember
		 WHERE user_no = #{userNo}
	</delete>

	<delete id="deleteAuth">
		DELETE FROM mybatismember_auth
		 WHERE user_no = #{userNo}
	</delete>

	<!-- 데이터베이스에서 기본키를 인수로 전달한 자바빈의 프로퍼티를 통해 취득한다. -->
	<!-- <insert id="create" useGeneratedKeys="true" keyProperty="userNo"> -->
	<insert id="create" parameterType="Member">
		<selectKey keyProperty="userNo" resultType="int" order="BEFORE">
		select mybatismember_seq.NEXTVAL FROM DUAL
		</selectKey>
		INSERT INTO mybatismember
			(
			  user_no,
			  user_id,
			  user_pw,
			  user_name
			)
		VALUES
			(
			  #{userNo},
			  #{userId},
			  #{userPw},
```

```
                        #{userName}
                    )
        </insert>

        <insert id="createAuth">
                INSERT INTO mybatismember_auth
                    (
                      user_no,
                      auth
                    )
                VALUES
                        (
                          #{userNo},
                          #{auth}
                        )
        </insert>

</mapper>
```

/src/main/webapp/WEB-INF/views/user

list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Mybatis 회원</title>
</head>
<body>
        <h3>회원 목록</h3>
        <a href="register">New</a>

        <table border="1">
                <tr>
                        <th align="center" width="60">NO</th>
                        <th align="center" width="80">USERID</th>
                        <th align="center" width="50">USERPW</th>
                        <th align="center" width="50">USERNAME</th>
                        <th align="center" width="180">REGDATE</th>
                </tr>
                <c:forEach items="${list}" var="member">
                        <tr>
                                <td align="center">${member.userNo}</td>
                                <td align="center"><a
href='/user/read?userNo=${member.userNo}'>${member.userId}</a></td>
                                <td align="left">${member.userPw}</td>
                                <td align="right">${member.userName}</td>
                                <td align="center"><fmt:formatDate pattern="yyyy-MM-dd
HH:mm" value="${member.regDate}" /></td>
                        </tr>
                </c:forEach>
```

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

read.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Mybatis 회원</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
        $(document).ready(function() {

                var formObj = $("#member");

                console.log(formObj);

                $("#btnModify").on("click", function()
                        { formObj.attr("action",
                        "/user/modify");formObj.attr("method",
                        "get"); formObj.submit();
                });

                $("#btnRemove").on("click", function()
                        { formObj.attr("action",
                        "/user/remove");formObj.submit();
                });

                $("#btnList").on("click", function()
                        {self.location = "/user/list";
                });

        });
</script>
</head>
<body>
        <h3>회원 상세 정보</h3>

        <form:form modelAttribute="member">

                <form:hidden path="userNo" />

                <table>
                        <tr>
                                <td>userid</td>
                                <td><form:input path="userId" readonly="true" /></td>

                        </tr>
                        <tr>
                                <td>username</td>
                                <td><form:input path="userName" readonly="true" /></td>
```

```
                    </tr>
                    <tr>
                            <td>auth - 1</td>
                            <td><form:select path="authList[0].auth">
                                        <form:option value="" label="=== 선택해 주세요 ===" />
                                        <form:option value="ROLE_USER" label="사용자" />
                                        <form:option value="ROLE_MEMBER" label="회원" />
                                        <form:option value="ROLE_ADMIN" label="관리자" />
                                    </form:select></td>
                    </tr>
                    <tr>
                            <td>auth - 2</td>
                            <td><form:select path="authList[1].auth">
                                        <form:option value="" label="=== 선택해 주세요 ===" />
                                        <form:option value="ROLE_USER" label="사용자" />
                                        <form:option value="ROLE_MEMBER" label="회원" />
                                        <form:option value="ROLE_ADMIN" label="관리자" />
                                    </form:select></td>
                    </tr>
                    <tr>
                            <td>auth - 3</td>
                            <td><form:select path="authList[2].auth">
                                        <form:option value="" label="=== 선택해 주세요 ===" />
                                        <form:option value="ROLE_USER" label="사용자" />
                                        <form:option value="ROLE_MEMBER" label="회원" />
                                        <form:option value="ROLE_ADMIN" label="관리자" />
                                    </form:select></td>
                    </tr>
            </table>
        </form:form>
        <div>
                <button type="submit" id="btnModify">Modify</button>
                <button type="submit" id="btnRemove">Remove</button>
                <button type="submit" id="btnList">List</button>
        </div>
</body>
</html>
```

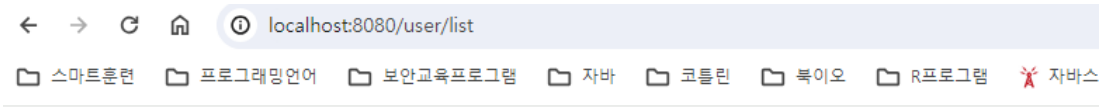
modify.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Mybatis 회원</title>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
	$(document).ready(function() {

		var formObj = $("#member");

		$("#btnModify").on("click", function()
			{ formObj.attr("action",
			"/user/modify");formObj.attr("method",
			"post"); formObj.submit();
		});

		$("#btnList").on("click", function()
			{self.location = "/user/list";
		});

	});
</script>
</head>
<body>
	<h3>회원 정보 수정</h3>

	<form:form modelAttribute="member" action="modify">

		<form:hidden path="userNo" />

		<table>
			<tr>
				<td>userid</td>
				<td><form:input path="userId" readonly="true" /></td>
				<td><font color="red"><form:errors path="userId"
/></font></td>
			</tr>
			<tr>
				<td>username</td>
				<td><form:input path="userName" /></td>
				<td><font color="red"><form:errors path="userName"
/></font></td>
		</tr>
		 <tr>
			<td>auth - 1</td>
			<td><form:select path="authList[0].auth">
					<form:option value="" label="=== 선택해주세요 ===" />
					<form:option value="ROLE_USER" label="사용자" />
					<form:option value="ROLE_MEMBER" label="회원" />
					<form:option value="ROLE_ADMIN" label="관리자" />
				</form:select></td>
			<td><font color="red"><form:errors
					path="authList[0].auth" /></font></td>
		</tr>
```

```
                        <tr>
                                <td>auth - 2</td>
                                <td><form:select path="authList[1].auth">
                                                <form:option value="" label="=== 선택해주세요 ===" />
                                                <form:option value="ROLE_USER" label="사용자" />
                                                <form:option value="ROLE_MEMBER" label="회원" />
                                                <form:option value="ROLE_ADMIN" label="관리자" />
                                        </form:select></td>
                                <td><font color="red"><form:errors
                                                        path="authList[1].auth" /></font></td>
                        </tr>

                        <tr>
                                <td>auth - 3</td>
                                <td><form:select path="authList[2].auth">
                                                <form:option value="" label="=== 선택해주세요 ===" />
                                                <form:option value="ROLE_USER" label="사용자" />
                                                <form:option value="ROLE_MEMBER" label="회원" />
                                                <form:option value="ROLE_ADMIN" label="관리자"/>
                                        </form:select></td>
                                <td><font color="red"><form:errors
                                                        path="authList[2].auth" /></font></td>
                        </tr>
                </table>
        </form:form>
        <div>
                <button type="submit" id="btnModify">Modify</button>
                <button type="submit" id="btnList">List</button>
        </div>
</body>
</html>
```

실행 결과

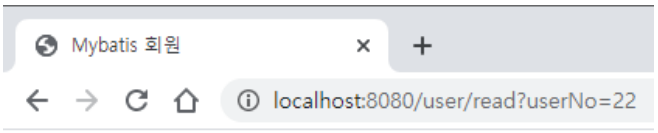
http://localhost:8080/user/list



회원 목록

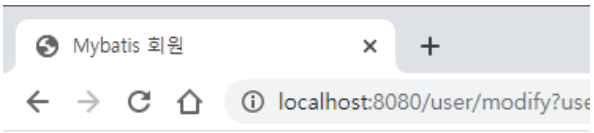
New

NO	USERID	USERPW	USERNAME	REGDATE
3	aaa	aaa	aaa	2023-12-31 01:29
1	jws	1234	제우스	2023-12-31 00:30



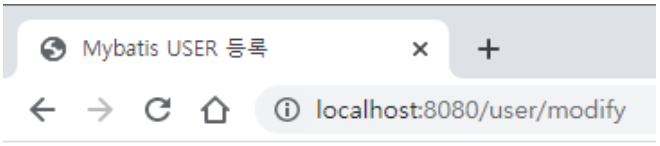
회원 상세 정보

userid jws
username 제우스
auth - 1 사용자
auth - 2 === 선택해 주세요 ===
auth - 3 === 선택해 주세요 ===
Modify Remove List

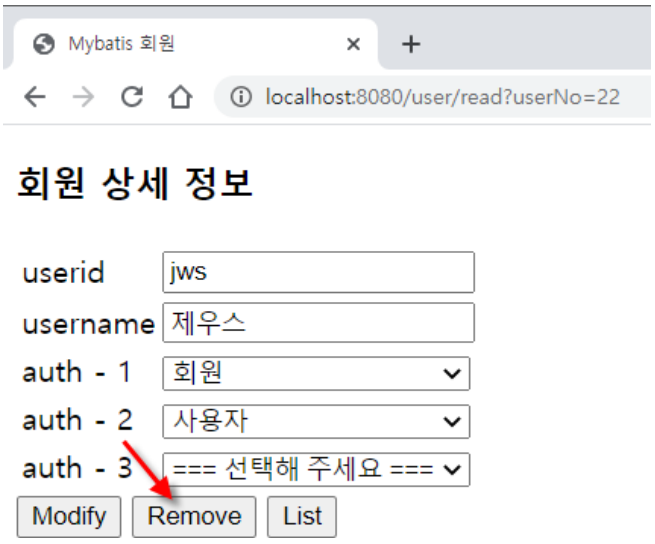


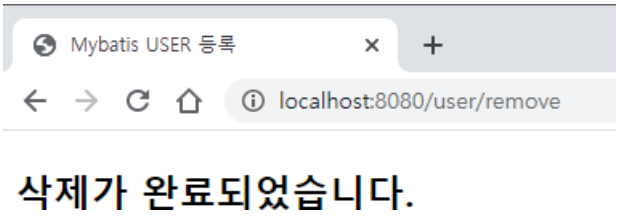
회원 정보 수정

userid jws
username 제우스
auth - 1 회원
auth - 2 사용자
auth - 3 === 선택해 주세요 ===
Modify List



수정이 완료되었습니다.


Mybatis 회원
localhost:8080/user/read?userNo=22
회원 상세 정보
userid
jws
username
제우스
auth - 1
회원
auth - 2
사용자
auth - 3
=== 선택해 주세요 ===
Modify
Remove
List


Mybatis USER 등록
localhost:8080/user/remove
삭제가 완료되었습니다.

14. 동적 SQL

MyBatis 의 가장 강력한 기능은 동적 SQL 기능이다. 동적 SQL 을 사용하는 것이 쉽지 않지만 MyBatis 는 매핑된 SQL 문에서 사용할 수 있는 강력한 동적 SQL 언어이다.

동적 SQL 요소

· if

· choose(when, otherwise)

· trim(where, set)

· foreach

(1) if

동적 SQL 에서 가장 일반적인 것은 조건부로 where 절의 일부를 포함하는 것이다.

```
<select id="findActiveBlog" resultType="Blog">
    SELECT * FROM blog
    WHERE state='active'
    <if test="title != null">
        AND title LIKE #{title}
    </if>
</select>
```

이 문장은 선택적으로 텍스트 검색 유형의 기능을 제공한다. title 을 전달하지 않으면 모든 활성 Blog 가 반환된다. 그러나 title 을 전달하면 비슷한 제목을 찾는다. (예를 들어, 이 경우 매개변수 값에는 마스킹 또는 와일드 카드 문자가 포함되어야 한다.)

title 과 author 별로 선택적으로 검색하려면 구문 이름을 알기 쉽게 변경한다. 그런 다음 조건을 추가한다.

```
<select id="findActiveBlog" resultType="Blog">
    SELECT * FROM blog
    WHERE state='active'
    <if test="title != null">
        AND title LIKE #{title}
    </if>
    <if test="author != null and author.name != null">
        AND author_name LIKE #{author.name}
    </if>
</select>
```

(2) choose, when, otherwise

경우에 따라 모든 조건을 적용하기를 원하지 않고 여러 옵션 중에서 하나의 사례만 선택하려고 한다. Java 의 switch 문과 유사하게 MyBatis 는 choose 요소를 제공한다. title 이 지정된 경우 제목만 검색하고 author 가 지정된 경우 저자만 검색한다. 둘 다 제공되지 않으면 추천 블로그만 반환한다.

```
<select id="findActiveBlog" resultType="Blog">
    SELECT * FROM blog
    WHERE state='active'
    <choose>
        <when test="title != null">
            AND title LIKE #{title}
        </when>
        <when test="author != null and author.name != null">
            AND author_name LIKE #{author.name}
        </when>
        <otherwise>
            AND featured = 1
        </otherwise>
    </choose>
</select>
```

(3) trim, where, set

위의 예제에서 동적 SQL 의 문제를 해결하는데 유용했다. if 예제에서 "state='active'"도 동적 조건으로 한다.

```
<select id="findActiveBlog" resultType="Blog">
    SELECT * FROM blog
    WHERE
      <if test="state != null">
            state=#{active}
    </if>
    <if test="title != null">
            AND title LIKE #{title}
    </if>
    <if test="author != null and author.name != null">
            AND author_name LIKE #{author.name}
    </if>
</select>
```

첫 번째 조건이 충족되지 않으면 다음과 같은 SQL 로 끝난다.

```
SELECT * FROM blog
WHERE
```

이것은 오류가 되고, 두 번째 조건만 충족하면 다음과 같다.

```
SELECT * FROM blog
WHERE
AND title AND 'someTitle'
```

이것도 오류이다. 이 문제는 조건문에서 쉽게 해결할 수 없다.

다음과 같이 간단한 방법으로 해결할 수 있다.

```
<select id="findActiveBlog" resultType="Blog">
    SELECT * FROM blog
    <where>
      <if test="state != null">
           state=#{active}
      </if>
      <if test="title != null">
           AND title LIKE #{title}
      </if>
      <if test="author != null and author.name != null">
           AND author_name LIKE #{author.name}
      </if>
    </where>
</select>
```

where 요소는 포함 태그에 의해 반환된 내용이 있는 경우에만 "WHERE"를 삽입한다. 또한 해당 컨텐츠가 "AND" 또는 "OR"로 시작하면 이를 제거한다.

where 요소가 원하는 대로 정확하게 작동하지 않으면 trim 요소를 정의할 수 있다.

```
<trim prefix="WHERE" prefixOverrides="AND | OR">
...
</trim>
```

prefixOverrides 속성은 파이프 문자로 구분된 텍스트 목록을 대체한다. 결과는 prefixOverrides 속성에 있는 것을 삽입한다.

set 이라는 동적 update 구문에 대한 비슷한 솔루션이 있다. set 요소를 사용하여 업데이트할 컬럼을 동적으로 포함하고 다른 컬럼을 생략할 수 있다.

```
<update id="updateAuthor">
    UPDATE author
       <set>
           <if test="username != null"> username=#{username}, </if>
```

```
        <if test="paswword != null"> password=#{password}, </if>
        <if test="email != null"> email=#{email}, </if>
        <if test="bio != null"> bio=#{bio} </if>
      </set>
    WHERE id=#{id}
</update>
```

여기서 set 요소는 SET 키워드를 동적으로 추가하고 조건이 적용된 후 값을 할당 수 있는 불필요한 쉼표를 제거한다.

```
<trim prefix="SET" suffixOverrides=",">
...
</trim>
```

(4) foreach

동적 SQL 에 대한 또 다른 공통적인 요구 사항은 종종 IN 조건을 빌드하기 위해 콜렉션을 반복해야 한다는 것이다.

```
<select id="selectPostIn" resultType="domain.blog.Post">
    SELECT *
    FROM post p
    WHERE id in
    <foreach item="item" index="index" collection="list" open="(" separator="," close=")">
        #{item}
    </foreach>
<select>
```

foreach 요소는 매우 강력하며 요소 본문 내에서 사용할 수 있는 컬렉션을 지정하고 항목을 선언하고 인덱스 변수를 지정할 수 있다. 또한 여는 문자열과 닫는 문자열을 지정하고 반복자 사이에 배치할 구분 기호를 추가할 수 있다. 이 요소는 실수로 여분의 구분 기호를 추가하지 않는다.

※ 참고
Iterable 객체(List, Set 등)와 Map 또는 Array 객체를 전달하여 컬렉션 매개변수로 foreach 할 수 있다. Iterable 또는 Array 를 사용하는 경우 index 는 현재 반복 횟수이고, value item 은 반복해서 검색된 요소이다. Map(또는 Map.Entry 객체의 컬렉션)을 사용하는 경우 index 는 주요 객체가 되고 item 은 값 객체가 된다.

15. 동적 SQL 적용

MyBatis 는 동적 SQL 을 조립하는 구조를 지원하고, SQL 조립 규칙을 매핑 파일에 정의할 수 있다.

(1) 동적으로 SQL 을 조립하기 위한 SQL 요소

① <whrer>

: WHERE 절 앞뒤에 내용을 더 추가하거나 삭제할 때 사용하는 요소

② <if>

: 조건을 만족할 때만 SQL 을 조립할 수 있게 만드는 요소

③ <choose>

: 여러 선택 항목에서 조건에 만족할 때만 SQL 을 조립할 수 있게 만드는 요소

④ <foreach>

: 컬렉션이나 배열에 대해 반복 처리를 하기 위한 요소

⑤ <set>

: SET 절 앞뒤에 내용을 추가하거나 삭제할 때 사용하는 요소

(2) 데이터베이스 테이블

① 게시판 테이블

```
CREATE TABLE
      mybatisboard(board_no
      NUMBER,
      title VARCHAR2(100) NOT NULL,
      content VARCHAR2(500) NULL,
      writer VARCHAR2(50) NOT NULL,
      reg_date DATE DEFAULT SYSDATE,
      PRIMARY KEY (board_no)
);

create sequence mybatisboard_seq
start with 1
```

(2) MyBatis 설정

/resource/

mybatis-config.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>

	<!-- mapUnderscoreToCamelCase 프로퍼티 값을 true로 지정 -->
	<settings>
		<setting name="mapUnderscoreToCamelCase" value="true" />
	</settings>

	<!-- TypeAlias로 맵핑 파일에서 반복적으로 사용될 패키지의 이름을 정의한다. -->
	<typeAliases>
		<package name="com.zeus.domain" />
	</typeAliases>

</configuration>
```

(3) MyBatis 구현

① Mapper 인터페이스

/src/main/java/com/zeus/mapper/

BoardMapper.java

```
package com.zeus.mapper;

import java.util.List;
import org.apache.ibatis.annotations.Param;
import com.zeus.domain.Board;

public interface BoardMapper {

	public void create(Board board) throws Exception;
	public Board read(Integer boardNo) throws Exception;
	public void update(Board board) throws Exception;
	public void delete(Integer boardNo) throws Exception;
	public List<Board> list() throws Exception;
	public List<Board> search(@Param("title") String title) throws Exception;

}
```

/src/main/resources/com/zeus/mapper/

BoardMapper.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="com.zeus.mapper.BoardMapper">
	<insert id="create">
		INSERT INTO mybatisboard
			(
			  board_no,
			  title,
			  content,
			  writer
			)
		VALUES
		    (
		      mybatisboard_seq.NEXTVAL,
		      #{title},
		      #{content},
		      #{writer}
		    )
	</insert>

	<!-- _로 구분된 컬럼명을 소문자 낙타표기법의 프로퍼티명으로 자동 매핑한다. -->
	<!-- 패키지명을 생략하고 클래스명만 지정할 수 있다. -->
	<select id="read" resultType="Board">
		SELECT board_no,
			   title,
			   content,
			   writer,
			   reg_date
		  FROM mybatisboard
		 WHERE board_no = #{boardNo}
	</select>

	<update id="update">
		UPDATE mybatisboard
		   SET title = #{title},
		       content = #{content}
		 WHERE board_no = #{boardNo}
	</update>

	<delete id="delete">
		DELETE FROM mybatisboard
		 WHERE board_no = #{boardNo}
	</delete>

	<!-- _로 구분된 컬럼명을 소문자 낙타표기법의 프로퍼티명으로 자동 매핑한다. -->
	<!-- 패키지명을 생략하고 클래스명만 지정할 수 있다. -->
	<select id="list" resultType="Board">
		<![CDATA[
		SELECT board_no,
			   title,
			   content,
			   writer,
			   reg_date
		  FROM mybatisboard
		 WHERE board_no > 0
	  ORDER BY board_no DESC,
			   reg_date DESC
```

```
            ]]>
        </select>

        <select id="search" resultType="Board">
            <![CDATA[
            SELECT board_no,
                       title,
                       content,
                       writer,
                       reg_date
              FROM mybatisboard
             WHERE board_no > 0
            ]]>

            <!-- title이 빈 값이 아니면 쿼리를 포함한다. -->
            <!-- CONCAT('', '') : 두 개의 문자 결합 -->
            <if test="title != null and title != ''">
               AND title LIKE CONCAT('%', CONCAT(#{title}, '%'))
            </if>

            <![CDATA[
            ORDER BY board_no DESC,
                        reg_date DESC
            ]]>
        </select>

</mapper>
```

(4) 게시판 구현 설명(추가된 내용만)

① 목록 화면에서 검색창 추가

· 요청

```
http://localhost:8080/board/list
```

· 컨트롤러 메서드

/src/main/java/com/zeus/controller

BoardController.java

```
package com.zeus.controller;

import org.mybatis.spring.annotation.MapperScan;
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 org.springframework.web.bind.annotation.RequestParam;

import com.zeus.domain.Board;
import com.zeus.service.BoardService;
```

```java
import lombok.extern.java.Log;

@Log
@Controller
@RequestMapping("/board")
@MapperScan(basePackages = "com.zeus.mapper")
public class BoardController {

	@Autowired
	private BoardService service;

	@RequestMapping(value = "/search", method = RequestMethod.POST)
	public String search(String title, Model model) throws Exception {
		log.info("search");
		Board board = new Board();
		board.setTitle(title);

		model.addAttribute("board", board);

		model.addAttribute("list", service.search(title));

		return "board/list";
	}

	@RequestMapping(value = "/register", method = RequestMethod.GET)
	public void registerForm(Board board, Model model) throws Exception {
		log.info("registerForm");
	}

	@RequestMapping(value = "/register", method = RequestMethod.POST)
	public String register(Board board, Model model) throws Exception {
		service.register(board);
		model.addAttribute("msg", "등록이 완료되었습니다.");
		return "board/success";
	}

	@RequestMapping(value = "/list", method = RequestMethod.GET)
	public void list(Model model) throws Exception {
		log.info("list");
		model.addAttribute("board", new Board());
		model.addAttribute("list", service.list());
	}

	@RequestMapping(value = "/read", method = RequestMethod.GET)
	public void read(@RequestParam("boardNo") int boardNo, Model model) throws
Exception {
		model.addAttribute(service.read(boardNo));
	}

	@RequestMapping(value = "/remove", method = RequestMethod.POST)
	public String remove(@RequestParam("boardNo") int boardNo, Model model) throws
Exception {
		service.remove(boardNo);
		model.addAttribute("msg", "삭제가 완료되었습니다.");
		return "board/success";
	}
```

```
        @RequestMapping(value = "/modify", method = RequestMethod.GET)
        public void modifyForm(int boardNo, Model model) throws Exception {
                model.addAttribute(service.read(boardNo));
        }

        @RequestMapping(value = "/modify", method = RequestMethod.POST)
        public String modify(Board board, Model model) throws Exception {
                service.modify(board);
                model.addAttribute("msg", "수정이 완료되었습니다.");
                return "board/success";
        }
}
```

· 서비스 인터페이스 메서드

/src/main/java/com/zeus/service

BoardService.java

```
package com.zeus.service;

import  java.util.List;
import com.zeus.domain.Board;

public interface BoardService {
        public void register(Board board) throws Exception;
        public Board read(Integer boardNo) throws Exception;
        public void modify(Board board) throws Exception;
        public void remove(Integer boardNo) throws Exception;
        public List<Board> list() throws Exception;
        public List<Board> search(String title) throws Exception;
}
```

BoardServiceImpl.java

```
package com.zeus.service;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.zeus.domain.Board;
import com.zeus.mapper.BoardMapper;

@Service
public class BoardServiceImpl implements BoardService {

        @Autowired
        private BoardMapper mapper;

        @Override
        public void register(Board board) throws Exception
                {mapper.create(board);
        }

        @Override
```

```
        public Board read(Integer boardNo) throws Exception
                {return mapper.read(boardNo);
        }

        @Override
        public void modify(Board board) throws Exception
                {mapper.update(board);
        }

        @Override
        public void remove(Integer boardNo) throws Exception
                {mapper.delete(boardNo);
        }

        @Override
        public List<Board> list() throws Exception
                {return mapper.list();
        }

        @Override
        public List<Board> search(String title) throws Exception
                {return mapper.search(title);
        }
}
```

· 뷰 파일

/src/main/webapp/WEB-INF/views/board

list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" session="false"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>MyBatis 게시판</title>
</head>
<body>
<form:form modelAttribute="board" method="POST" action="search">
        <h2>게시글 목록</h2>
        TITLE : <form:input path="title" /><input type="submit" value="검색" />
        <a href="register">New</a>
        <table border="1">
                <tr>
                        <th align="center" width="80">NO</th>
                        <th align="center" width="320">TITLE</th>
                        <th align="center" width="100">WRITER</th>
                        <th align="center" width="180">REGDATE</th>
                </tr>
                <c:choose>
                        <c:when test="${empty list}">
```

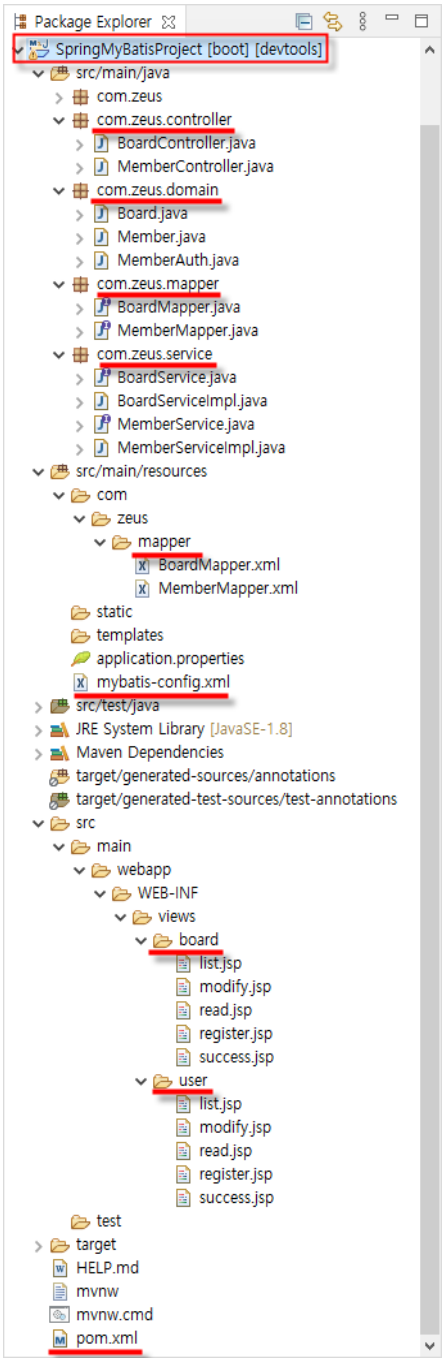
```
                        <tr>
                                <td colspan="4">
                                        게시글이 없습니다.
                                </td>
                        </tr>
                </c:when>
                <c:otherwise>
                        <c:forEach items="${list}" var="board">
                                <tr>
                                        <td align="center">${board.boardNo}</td>
                                        <td align="left"><a
href='/board/read?boardNo=${board.boardNo}'>${board.title}</a></td>
                                        <td align="right">${board.writer}</td>
                                        <td align="center"><fmt:formatDate
pattern="yyyy-MM-dd HH:mm" value="${board.regDate}" /></td>
                                </tr>
                        </c:forEach>
                </c:otherwise>
        </c:choose>
    </table>
</form:form>
</body>
</html>
```

실행 결과

http://localhost:8080/board/list

http://localhost:8080/board/search

전체 파일 구조


Package Explorer
SpringMyBatisProject [boot] [devtools]
src/main/java
com.zeus
com.zeus.controller
BoardController.java
MemberController.java
com.zeus.domain
Board.java
Member.java
MemberAuth.java
com.zeus.mapper
BoardMapper.java
MemberMapper.java
com.zeus.service
BoardService.java
BoardServiceImpl.java
MemberService.java
MemberServiceImpl.java
src/main/resources
com
zeus
mapper
BoardMapper.xml
MemberMapper.xml
static
templates
application.properties
mybatis-config.xml
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
target/generated-sources/annotations
target/generated-test-sources/test-annotations
src
main
webapp
WEB-INF
views
board
list.jsp
modify.jsp
read.jsp
register.jsp
success.jsp
user
list.jsp
modify.jsp
read.jsp
register.jsp
success.jsp
test
target
HELP.md
mvnw
mvnw.cmd
pom.xml