4. Mybatis

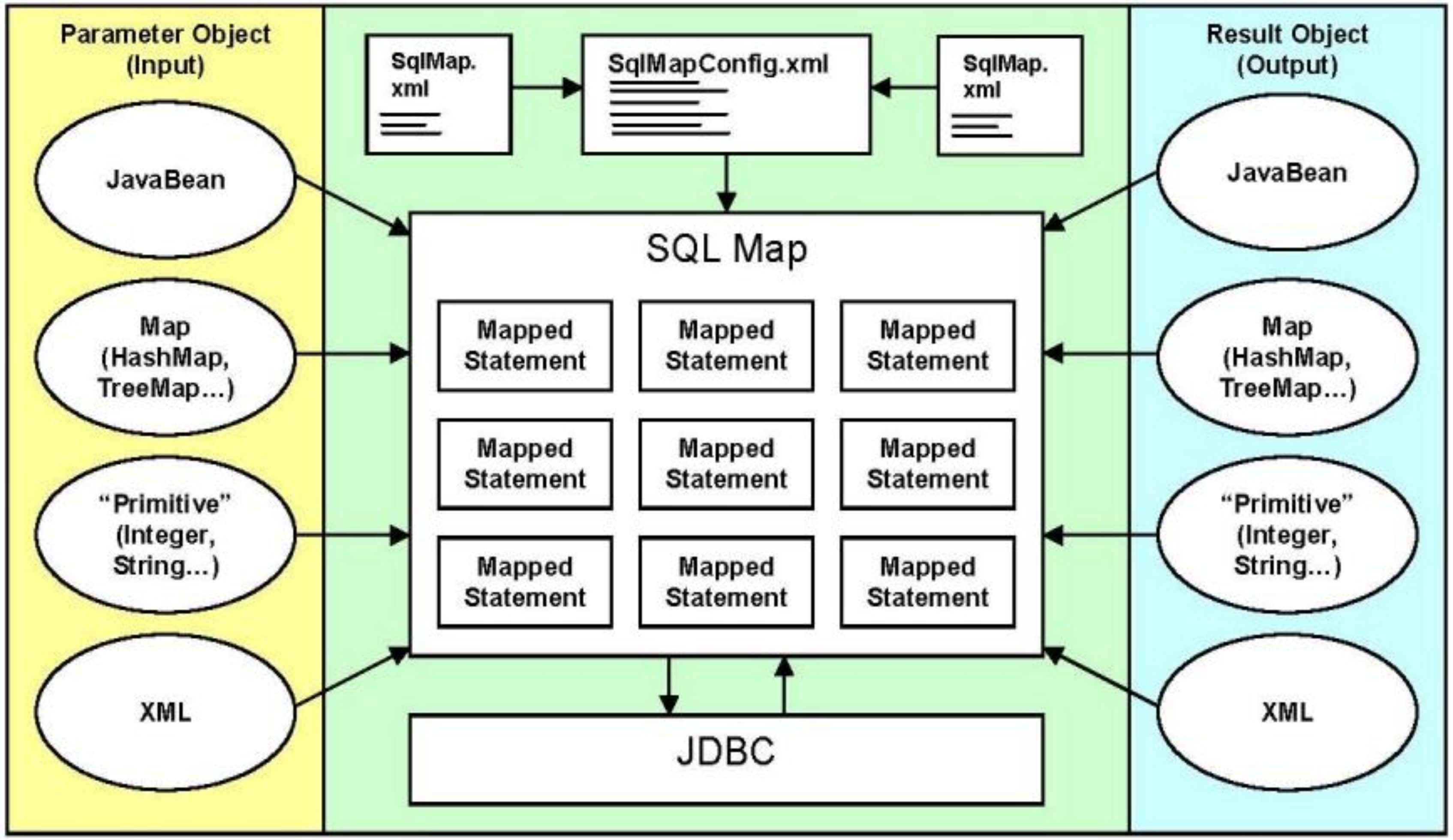
I JDBC

```java
public void insert(EmpVO vo) {
  try {
    //1. connect
    conn = ds.getConnection();
    //트랜잭션 범위 시작
    conn.setAutoCommit(false);
    //2. statement
    String sql = "INSERT INTO EMPLOYEES values ( ?,?,?)";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, vo.getEmployeeNo());
    pstmt.setString(2, vo.getFirstName());
   pstmt.setString(3, vo.getLastName());
    pstmt.executeUpdate();
    sql = "INSERT INTO login values(?,?) ";
    pstmt = conn.prepareStatement(sql);
    int r = pstmt.executeUpdate();
    //커밋 : 트랜잭션 범위 종료
    conn.commit();
  } catch(Exception e){
    if(conn != null)
      try { conn.rollback(); } catch (SQLException e1) { }
  } finally {
    if(conn != null) try { conn.close(); } catch (SQLException e1) { }
  }
}
```

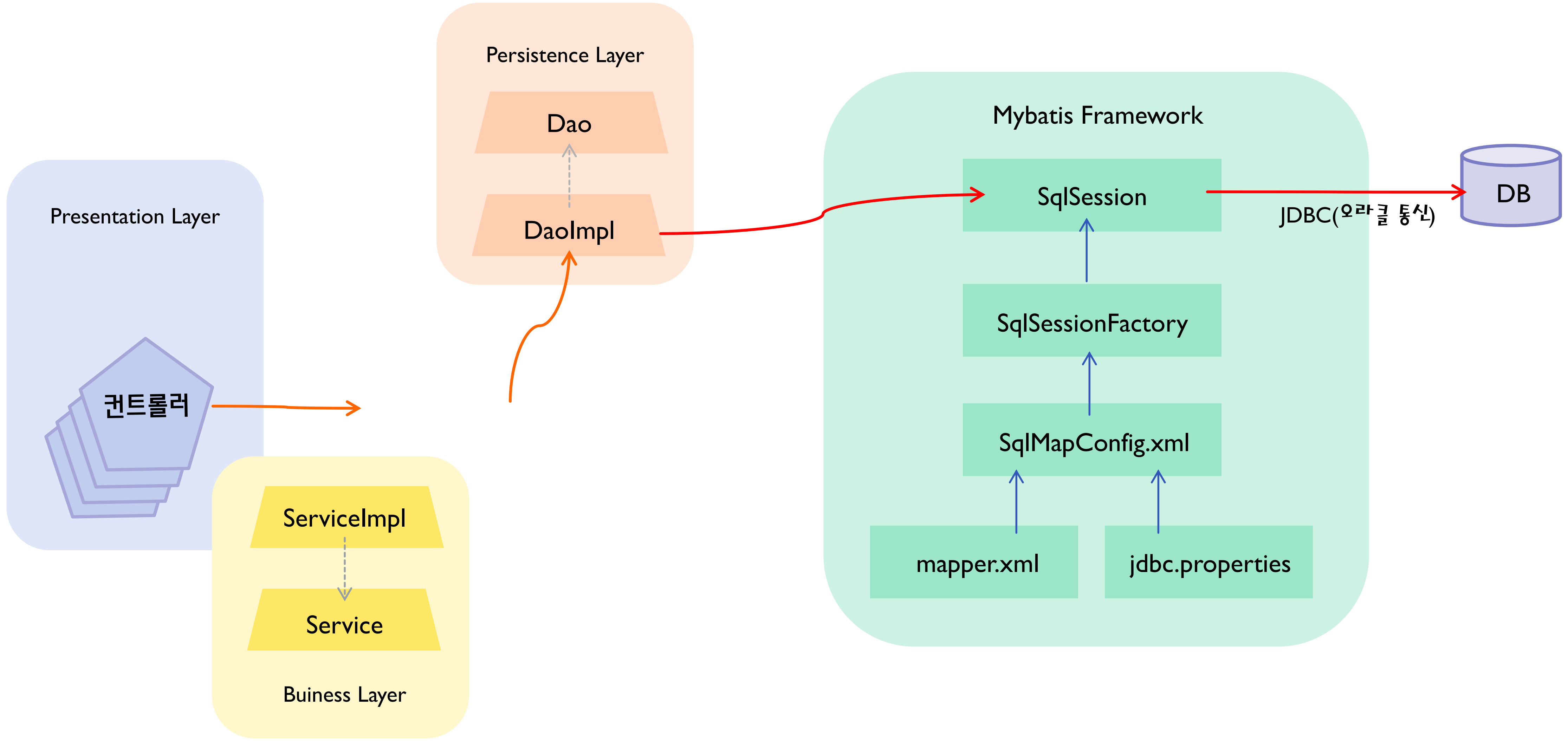
2.1 Mybatis

- JDBC의 모든 기능을 MyBatis가 대부분 제공하므로 한 두 줄의 자바 코드로 DB 연동을 처리.
- SQL 명령어를 자바코드에서 분리하여 XML로 따로 관리
- XML 파일에 저장된 SQL 명령어를 대신 실행하고 실행결과를 VO 같은 자바 객체에 자동으로 매핑까지 해준다.

2.2 Mybatis 구성



2.3 Mybatis 주요 컴포넌트



3.1 mybatis 설정

- sql-map-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>
<settings>
  <setting name="jdbcTypeForNull" value="VARCHAR"/>
  <setting name="mapUnderscoreToCamelCase" value="true"/>
</settings>
</configuration>
<!-- Alias 설정 -->
<typeAliases>
   <typeAlias alias="board" type="com.springbook.biz.board.BoardVO"/>
</typeAliases>
<typeHandlers>
 <!-- java.sql.Timestamp 를 java.util.Date 형으로 반환 -->
 <typeHandler javaType="java.sql.Date" handler="org.apache.ibatis.type.DateTypeHandler"/>
</typeHandlers>
```

3.1 커넥션 풀 설정

- pom.xml 에 <dependency> 추가
  - HikariCP
  - spring-jdbc (spring-tx 포함됨)
  - ojdbc8

```xml
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <springframework.version>6.2.7</springframework.version>
</properties>

<!-- Database connection pool -->
<dependency>
  <groupId>com.zaxxer</groupId>
  <artifactId>HikariCP</artifactId>
  <version>6.3.0</version>
</dependency>

<!-- spring-jdbc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>${org.sprigframework-version}</version>
</dependency>

<!-- ojdbc8 -->
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc8</artifactId>
  <version>23.2.0.0</version>
</dependency>
```

3.1 커넥션 테스트

```
package com.yedam.web;

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

public class ConnectionTest {
     static {
          try {
               Class.forName("oracle.jdbc.driver.OracleDriver");
          } catch(Exception e) {
               e.printStackTrace();
          }
     }

     @Test
     public void testConnection() {
          try(Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","hr","hr")) {
               System.out.println(con);
          }catch(Exception e) {
               System.out.println(e.getMessage());
          }
     }
}
```

3.2 Mybatis 설정

- src\main\resources\spring\datasource-context.xml

```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:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- datasource connection pool -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"   value="hr" />
  <property name="password"   value="hr" />
</bean>
<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>
```

3.2 datasource 테스트

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/resources/datasource.xml")
public class DataSourceTest {

    @Setter(onMethod_ = {@Autowired})
    private DataSource dataSource;

    @Setter(onMethod_ = {@Autowired})
    private SqlSessionFactory sqlSessionFactory;

    @Test
    public void testConnection() {
       try(Connection con = dataSource.getConnection()) {
           System.out.println(con);
      } catch(Exception e) {
           fail(e.getMessage());
      }
   }
}
```

2.2 Mybatis 설정

- pom.xml 에 <dependency> 추가
  - mybatis-spring
  - mybatis

```
<!-- mybatis -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.5.19</version>
</dependency>

<!-- mybatis-spring -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis-spring</artifactId>
<version>3.0.4</version>
</dependency>
```

2.2 Mybatis 설정

- src\main\resources\spring\datasource-context.xml

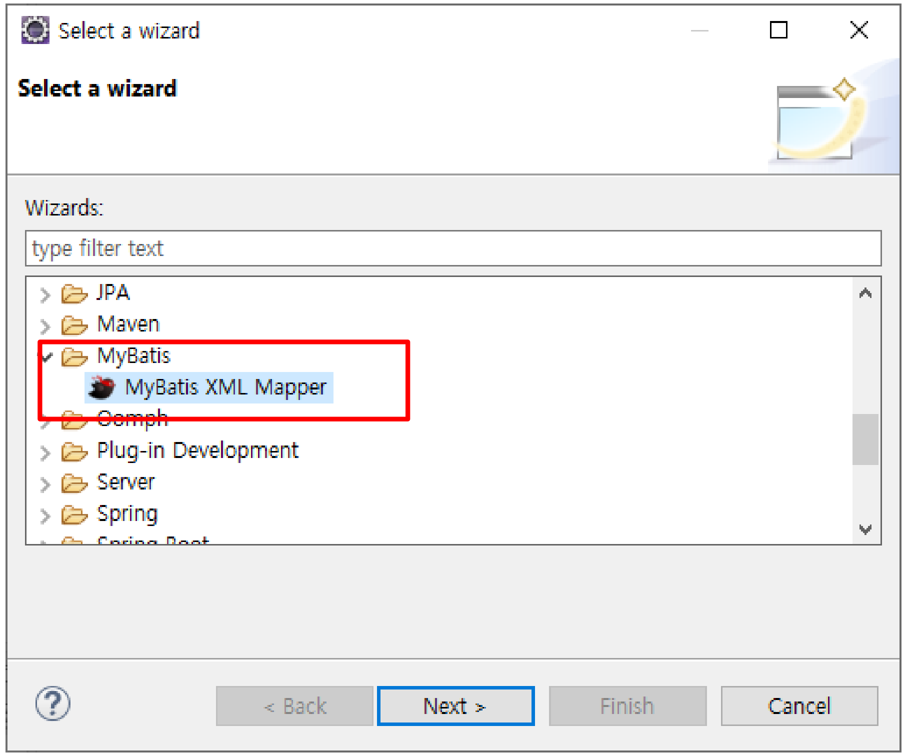
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- datasource connection pool -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"   value="hr" />
  <property name="password"   value="hr" />
</bean>
<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>

<!-- mybatis  SqlSessionFactory -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
    <property name="mapperLocations" value="classpath:mapper/*.xml" />
</bean>
<mybatis-spring:scan base-package="com.yedam.**.mapper"></mybatis-spring:scan>
</beans>
```

3.2 sqlSessionFactory 테스트

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("file:src/main/resources/datasource.xml")
public class DataSourceTest {

    @Setter(onMethod_ = {@Autowired})
    private DataSource dataSource;

    @Setter(onMethod_ = {@Autowired})
    private SqlSessionFactory sqlSessionFactory;

    @Test
    public void testMybatis() {
       try(SqlSession session = sqlSessionFactory.openSession(); Connection con = 
session.getConnection();){
           System.out.println(session);
       } catch(Exception e) {
          fail(e.getMessage());
       }
     }
}
```

4.1 Sql statement xml 파일 생성

- File -> New -> Other...



Select a wizard

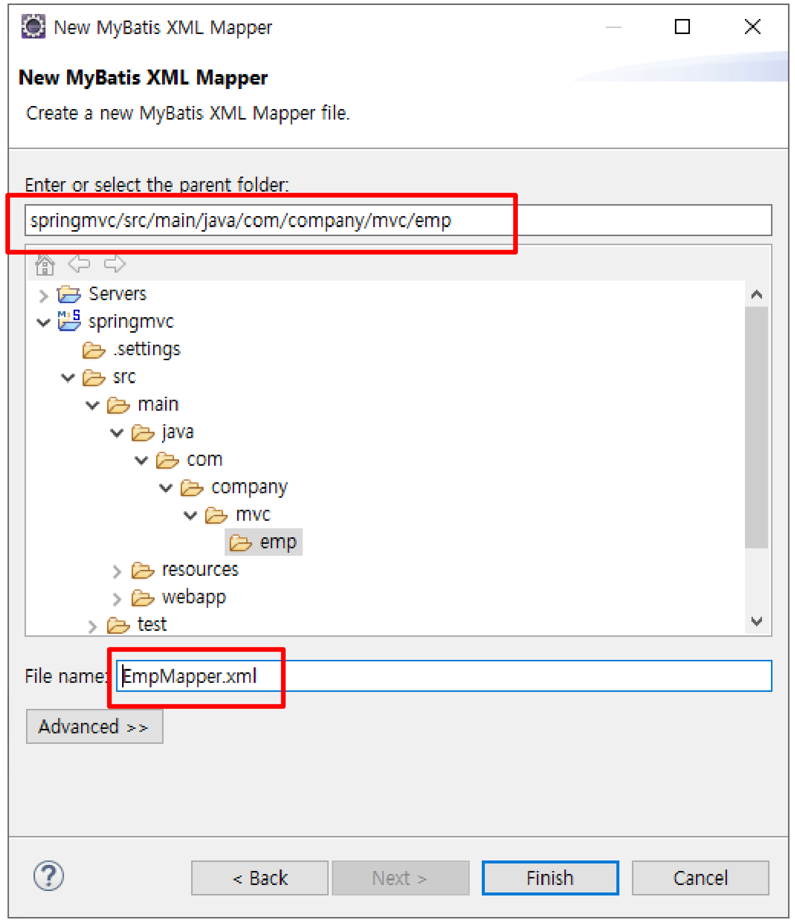
Select a wizard

Wizards:

type filter text

- JPA
- Maven
- MyBatis
  - MyBatis XML Mapper
- Oomph
- Plug-in Development
- Server
- Spring
- Spring Boot

- 생성위치와 파일명 입력



New MyBatis XML Mapper

New MyBatis XML Mapper

Create a new MyBatis XML Mapper file.

Enter or select the parent folder:

springmvc/src/main/java/com/company/mvc/emp

- Servers
- springmvc
  - .settings
  - src
    - main
      - java
        - com
          - company
            - mvc
              - emp
      - resources
      - webapp
    - test

File name: EmpMapper.xml

4.2 Sql statement xml 파일 생성

▪ EmpVO

```
@Data
public class EmpVO {
     String employee_id;
     String first_name;
     String last_name;
     String email;
     String hire_date;
     String job_id;
     String department_id;
     String salary;
}
```

▪ mapper 인터페이스

```
public interface EmpMapper {
   public EmpVO getEmp(EmpVO empVO);
}
```

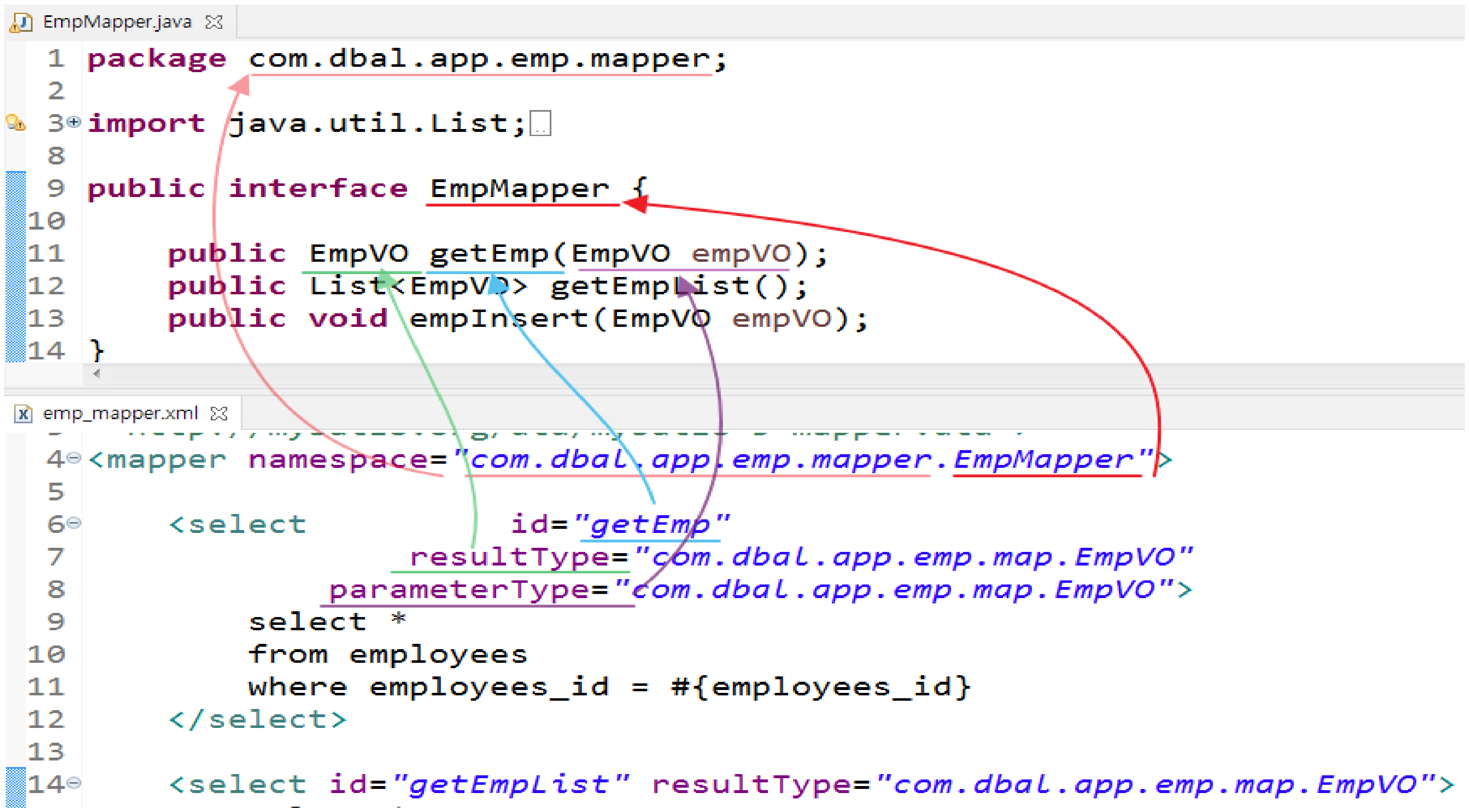
▪ sql statmement xml 파일

```
<?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.company.mvc.emp.EmpMapper">
<select id="getEmp"
        parameterType="com.company.mvc.emp.EmpVO"
        resultType="com.company.mvc.emp.EmpVO">
SELECT  employee_id,
        first_name,
        last_name,
        email,
        hire_date,
        job_id,
        salary
  FROM employees
 WHERE employee_id = #{employee_id}
</select>
</mapper>
```

4.3 DAO

```
package com.dbal.app.emp.mapper;
import java.util.List;
import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
@Repository
public class EmpDAOImpl  implements EmpDAO {
	@Autowired private SqlSessionTemplate mybatis;

	public EmpVO getEmp(EmpVO empVO) {
		return mybatis.selectOne("com.dbal.app.emp.map.EmpMapper.getEmp",empVO );
	}

	public List<EmpVO> getEmpList() {
		return mybatis.selectList("com.dbal.app.emp.map.EmpMapper.getEmpList");
	}
	public void empInsert(EmpVO empVO) {
		mybatis.insert("com.dbal.app.emp.map.EmpMapper.empInsert", empVO);
	}
}
```

4.4 인터페이스



```java
package com.dbal.app.emp.mapper;

import java.util.List;

public interface EmpMapper {

	public EmpVO getEmp(EmpVO empVO);
	public List<EmpVO> getEmpList();
	public void empInsert(EmpVO empVO);
}
```

```xml
<mapper namespace="com.dbal.app.emp.mapper.EmpMapper">

	<select			id="getEmp"
				resultType="com.dbal.app.emp.map.EmpVO"
			parameterType="com.dbal.app.emp.map.EmpVO">
		select *
		from employees
		where employees_id = #{employees_id}
	</select>

	<select id="getEmpList" resultType="com.dbal.app.emp.map.EmpVO">
```

4.5 동적 SQL

```
<!-- 전체조회  -->
<select id="getEmpList" resultType="emp">
    SELECT rownum id, e.*
      FROM employees e
    <where>
      <if test="departmentId != null"> department_id = #{departmentId} </if>
      <if test="employeeId != null"> and employee_id = #{employeeId} </if>
      <if test="salary != null">  <![CDATA[ and salary < #{salary} ]]> </if>
      <if test="firstName != null">
           and upper(first_name) like '%' || upper(#{firstName}) || '%'
      </if>
      <if test="employeeIds != null">  and employee_id in
        <foreach collection="employeeIds" item="id" open="(" close=")" separator=",">
            #{id}
        </foreach>
      </if>
     </where>
     ORDER BY first_name
</select>
```

4.6 트랜잭션

- datasource-context.xml

```
<!-- TransactionManager bean 등록 -->
<bean id="txManager"
  class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>

<!-- @Transactional 어노테이션 처리 -->
<tx:annotation-driven transaction-manager="transactionManager" />
```

- @transactional

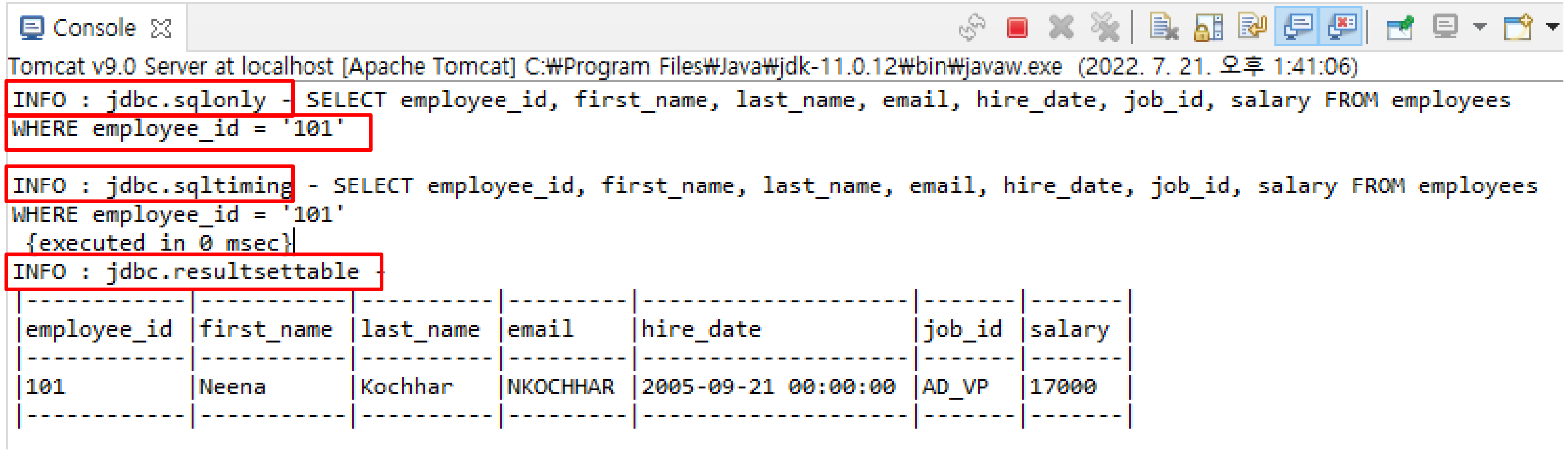
4.7 프로시저

```
<parameterMap type="board" id="boardParam">
        <parameter property="title" mode="IN" jdbcType="VARCHAR" javaType="string"/>
        <parameter property="writer" mode="IN" jdbcType="VARCHAR" javaType="string"/>
        <parameter property="content" mode="IN" jdbcType="VARCHAR" javaType="string"/>
        <parameter property="seq" mode="OUT" jdbcType="NUMERIC" javaType="int"/>
        <parameter property="out_msg" mode="OUT" jdbcType="VARCHAR" javaType="string"/>
</parameterMap>

<insert id="insertBoardProc1" statementType="CALLABLE" parameterMap="boardParam">
        { call BOARD_INS_PROC(?,?,?,?,?) }
</insert>
```

```
<insert id="insertBoardProc2" statementType="CALLABLE" parameterType="board">
        { call BOARD_INS_PROC(
         #{title},
         #{writer},
         #{content, mode=IN, jdbcType=VARCHAR, javaType=string},
         #{seq, mode=OUT, jdbcType=NUMERIC, javaType=java.math.BigDecimal},
         #{out_msg, mode=OUT, jdbcType=VARCHAR, javaType=string}
         ) }
</insert>
```

5.1 sql 로그 보기

- PreparedStatement에서 파라미터가 대입된 쿼리 내용과 실행결과를 볼 수 있다.



```
Tomcat v9.0 Server at localhost [Apache Tomcat] C:₩Program Files₩Java₩jdk-11.0.12₩bin₩javaw.exe  (2022. 7. 21. 오후 1:41:06)
INFO : jdbc.sqlonly - SELECT employee_id, first_name, last_name, email, hire_date, job_id, salary FROM employees
WHERE employee_id = '101'

INFO : jdbc.sqltiming - SELECT employee_id, first_name, last_name, email, hire_date, job_id, salary FROM employees
WHERE employee_id = '101'
 {executed in 0 msec}
INFO : jdbc.resultsettable -
|------------|-----------|----------|---------|--------------------|-------|-------|
|employee_id |first_name |last_name |email    |hire_date           |job_id |salary |
|------------|-----------|----------|---------|--------------------|-------|-------|
|101         |Neena      |Kochhar   |NKOCHHAR |2005-09-21 00:00:00 |AD_VP  |17000  |
|------------|-----------|----------|---------|--------------------|-------|-------|
```

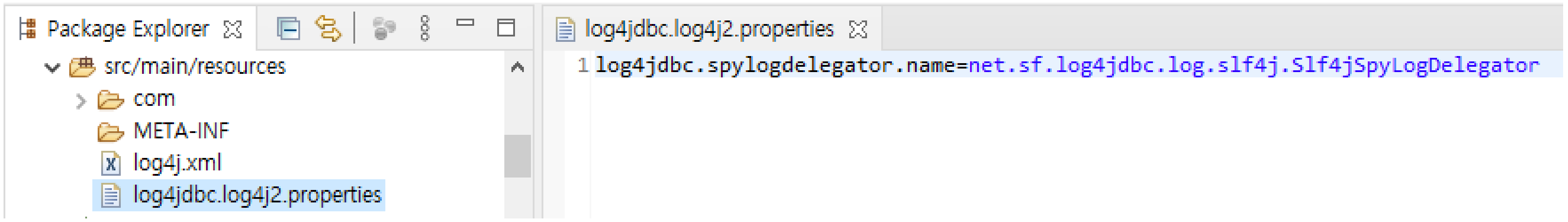
5.2 dependency 추가

- pom.xml
  - log4jdbc-log4j2 라이브러리 추가

```
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4.1</artifactId>
  <version>1.16</version>
</dependency>
```

- 로그 설정파일 추가
  - log4jdbc.log4j2.properties

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```



```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```

5.3 로그 레벨 설정

- src/main/resources/log4j2.xml
  - 로그 레벨
    - trace < debug < info < warn < error < fetal
    - 지정된 레벨 이하는 출력 안됨
    - off 는 로는 출력 안함
  - 루트 로그 레벨 설정
    - 패키지별 별도 지정이 없으면 루트 레벨을 적용함

```
<Root level="info" >
  <AppenderRef ref="Console" />
</Root>
```

- 패키지별 로그 레벨 설정

```
<Logger name="jdbc.sqlonly" level="info" additivity="false">
  <AppenderRef ref="Console" />
</Logger>

<Logger name=" jdbc.sqltiming" level="info" additivity="false">
  <AppenderRef ref="Console" />
</Logger>

<Logger name=" jdbc.resultsettable" level="info" additivity="false">
  <AppenderRef ref="Console" />
</Logger>

<Logger name=" jdbc.audit" level="info" additivity="false">
  <AppenderRef ref="Console" />
</Logger>

<Logger name=" jdbc.resultset" level="info" additivity="false">
  <AppenderRef ref="Console" />
</Logger>
```

5.4 datasource 변경

- JDBC 드라이버와 URL 정부 수정
  - src/main/resources/spring/datasource-context.xml
  - driverClassName 속성과 jdbcUrl 속성을 변경

```
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
<!-
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
-->
  <property name="driverClassName" value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy" />
  <property name="jdbcUrl"    value="jdbc:log4jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"        value="hr" />
  <property name="password"        value="hr" />
</bean>
```