myBatis

myBatis

- ✓ myBatis의 정의 및 특성
- ✓ myBatis API 및 SQL 관련 XML 태그
- ✓ 실습

myBatis 정의 및 특성

- ✓ 2010년에 iBatis가 Apache를 떠나 Google Code 로 이동하면서 이름이 myBatis로 변경됨.
- ✓ 더 빠른 JDBC 코딩을 위한 일반화된 프레임워크.
- ✓ SQL을 자바 코드가 아닌 XML로 따로 분리
- ✓ SQL의 실행 결과를 Map 또는 자바 클래스로 자동 매핑
- ✓ SQL을 XML 이나 인터페이스내에 어노테이션을 활용하여 처리

Jdbc 프로그래밍 코드의 번거로움

- ✓ 1. DBMS 드라이버 로드
- ✓ 2. DB 연결
- ✓ 3. SQL문 DB로 송신
- ✓ 4. SQL 결과 수신
- ✓ 5. 연결닫기

➔ SQL 만 작성하고 싶다

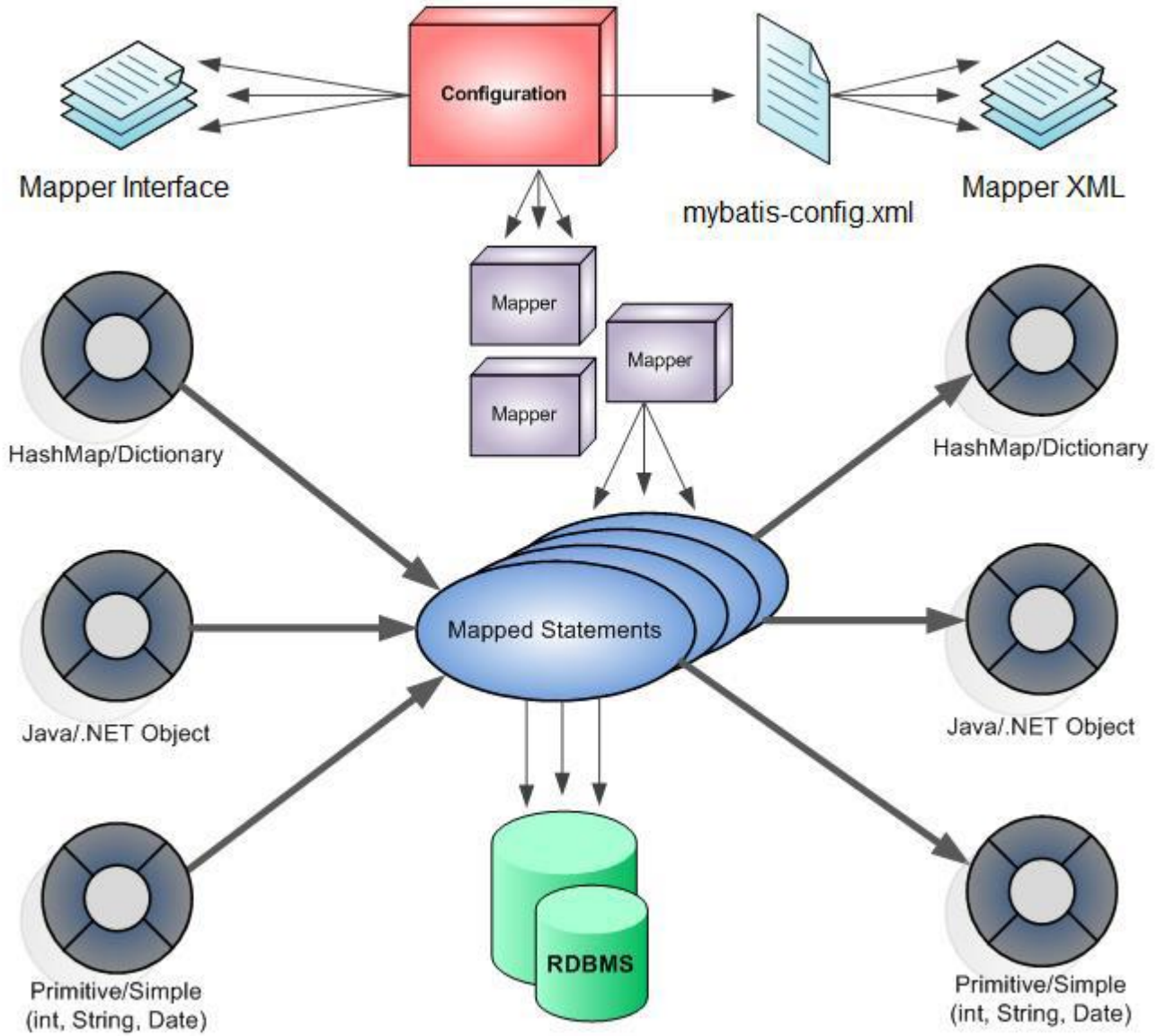
```
//1. DBMS 드라이버 로드
Class.forName("oracle.jdbc.driver.OracleDriver");

//2. DB 연결
Connection con = null;
String url = "jdbc:oracle:thin:@localhost:1521:XE";
String user = "USER";
String password = "PASSWORD";
con = DriverManager.getConnection(url, user, password);

//3. SQL문 DB로 송신
String sql = "SELECT * FROM MYTABLE";
PreparedStatement pstmt = null;
pstmt = con.prepareStatement(sql);

//4. SQL 결과 수신
ResultSet rs = null;
rs = pstmt.executeQuery();
ArrayList<Customer> list = new ArrayList<>();
while(rs.next()){
    String id = rs.getString("ID");
    String pwd = rs.getString("PASSWORD");
    String name = rs.getString("NAME");
    String email = rs.getString("EMAIL");
    Customer customer = new Customer(id, pwd, name, email);
    list.add(customer);
}
//5. 연결닫기
rs.close();
pstmt.close();
con.close();
```

myBatis 정의 및 특성



myBatis 적용

- ✓ 환경설정 파일(SqIMapConfig)과 실제 쿼리를 적용할 파일 (Mapper)이 필요(XML, Annotation)
- ✓ 자바파일에서는 설정된 환경설정 정보를 이용하여 SqISession객체를 얻어온 다음 myBatis와 연동

myBatis 다운로드

- ✓ https://github.com/mybatis/mybatis-3/releases 다운로드
- ✓ 프로젝트에 다운받은 파일을 압축 해재 한 후 mybatis-x.x.x.jar 파일을 클래스패스에 추가(웹 프로젝트인 경우 WEB-INF/lib에 추가)
- ✓ Maven 일 경우 Dependency 추가

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>x.x.x</version>
</dependency>
```

xml 환경 설정

✓ 설정 루트

```
<configuration>
   - vo 객체의 타입 Alias 설정
   <typeAliases>
      <typeAlias alias="xxx" type="xxx.Xxx" />
   </typeAliases>
   - 설정파일 내부에서 사용하는 값을 파일을 활용하여 추출
   <properties resource="db.properties" />
   …… 생략…
</configuration>
```

xml 환경 설정

```
<configuration>
    …… 생략…
    - 트랜잭션 및 데이터베이스 설정
    <environments default="development">
      <environment id="development">
        <transactionManager type="JDBC"/>
        <dataSource type="POOLED">
           <property name="driver" value="…"/>
           <property name="url" value="…"/>
           <property name="username" value="hr"/>
           <property name="password" value="hr"/>
        </dataSource>
      </environment>
    </environments>
</configuration>
```

트랜잭션

UNPOOLED, POOLED, JNDI

데이터베이스

xml 환경 설정

```
<environments default="development">
 <environment id="development">
  <transactionManager type="JDBC"/>
   <dataSource type="POOLED">
    <property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
    <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe"/>
    <property name="username" value="c##heejin"/>
    <property name="password" value="heejin"/>
   </dataSource>
  </environment>
</environments>
```

xml 환경 설정 – oracle xe

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:xe
username=c##heejin
password=heejin
```

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

<environments default="development">
 <environment id="development">
  <transactionManager type="JDBC"/>
   <dataSource type="POOLED">
    <property name="driver" value="${driver}"/>
    <property name="url" value="${url}"/>
    <property name="username" value="${username}"/>
    <property name="password" value="${password}"/>
   </dataSource>
  </environment>
</environments>
```

xml 환경 설정 – oracle cloud

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:[tnsName]?TNS_ADMIN=[월렛디렉토리]
username=c##heejin
password=heejin
```

예시

```
url=jdbc:oracle:thin:@db202110031651_high?TNS_ADMIN
=./src/main/resources/Wallet/
```

xml 환경 설정

- 매퍼정보 설정

```
    <mappers>
        <mapper resource="common/db/xxx.xml"/>
    </mappers>
</configuration>
```

XML Mapper 사용 태그

구문형식	속성	하위 요소	사용 용도
<select>	id parameteType resultType resultMap	모든 동적 요소	데이터 조회
<sql>	id		공통 쿼리 묶기
<include>	refid		<sql> 로 생성된 컴포넌트 삽입
<insert>	id parameterType	모든 동적 요소 <selectKey>	데이터 입력
<update>	id parameterType	모든 동적 요소	데이터 수정
<delete>	id parameterType	모든 동적 요소	데이터 삭제

sql / include

구문형식	속성	하위 요소	사용 용도
<sql>	id		공통 쿼리 묶기
<include>	refid		<sql> 로 생성된 컴포넌트 삽입

```
<sql id="select_t_board">
  select * from t_board
</sql>
```

```
<select id="selectByNo">
  select * from t_board
   where no=3
</select>
```

```
<select id="selectByNo">
  <include refid="select_t_board"/>
   where no=3
</select>
```

```
<select id="selectById">
  select * from t_board
   where id='hong'
</select>
```

```
<select id="selectById">
  <include refid="select_t_board"/>
   where id='hong'
</select>
```

XML Mapper 동적 태그

구문형식	속성	사용법
<foreach>	Collection Item Open Separator close	<foreach item="addr" open="(" separator="," close=")" collection="addrs"> #{addr} </foreach>
<if>	test	<if test="id != null">id = #{id}</if>
<choose> <when> <otherwise>	test	<choose> <when test="searchType == 'id'"> id like #{searchWord} </when> <otherwise> addr like #{searchWord} </otherwise> </choose>

XML Mapper 동적 태그

구문형식	속성	사용법
<where>		<where> <if test="writer != null"> writer like #{writer} </if> <if test="title != null"> AND title like #{title} </if> </where>
<set>		<set> <if test="writer != null"> writer = #{writer}, </if> <if test="title != null"> title = #{title} </if> </set>

foreach

구문형식	속성	사용법
<foreach>	Collection Item Open Separator close	<foreach item="addr" open="(" separator="," close=")" collection="addrs"> #{addr} </foreach>

```
select * from t_board
 where no in (1,2,3,4,5,6)
```

→ numbers

```
select * from t_board
  where no in
```

```
<foreach collections="numbers" item="number"
         separator="," start ="(", end=")">
   #{number}
</foreach>
```

if

구문형식	속성	사용법
<if>	test	<if test="id != null">id = #{id}</if>

기본적으로는 전체 값을 search하고,
만약에 id를 값으로 받는다면, id로 조회해라

```
select * from t_board
```

```
select * from t_board
  where id=3
```

```
select * from t_board
 <if test="id != null"> where id=#{id} </if>
```

choose

구문형식	속성	사용법
<choose> <when> <otherwise>	test	<choose> <when test="searchType == 'id'"> id like #{searchWord} </when> <otherwise> addr like #{searchWord} </otherwise> </choose>

Id가 들어올수도 있고, addr가 들어올수도 있는 경우, 한번에 쓰고 싶을때, type에 따라서 바뀐다.

```
select * from t_board
  where id like ‘hong’
```

```
select * from t_board
  where addr like ‘seoul’
```

```
select * from t_board
  where
    <choose>
          <when test = “searchType == ‘id’ ”>
              id like #{searchWord}
          </when>
          <otherwise>
              addr like #{searchWord}
          </otherwise>
    </choose>
```

where

구문형식	속성	사용법
<where>		<where> <if test="writer != null"> writer like #{writer} </if> <if test="title != null"> AND title like #{title} </if> </where>

게시판에서 검색시 검색 방법을 write의 이름을 넣을수도 있고, title을 넣을수도 있고, 둘다 넣을수도 있다.

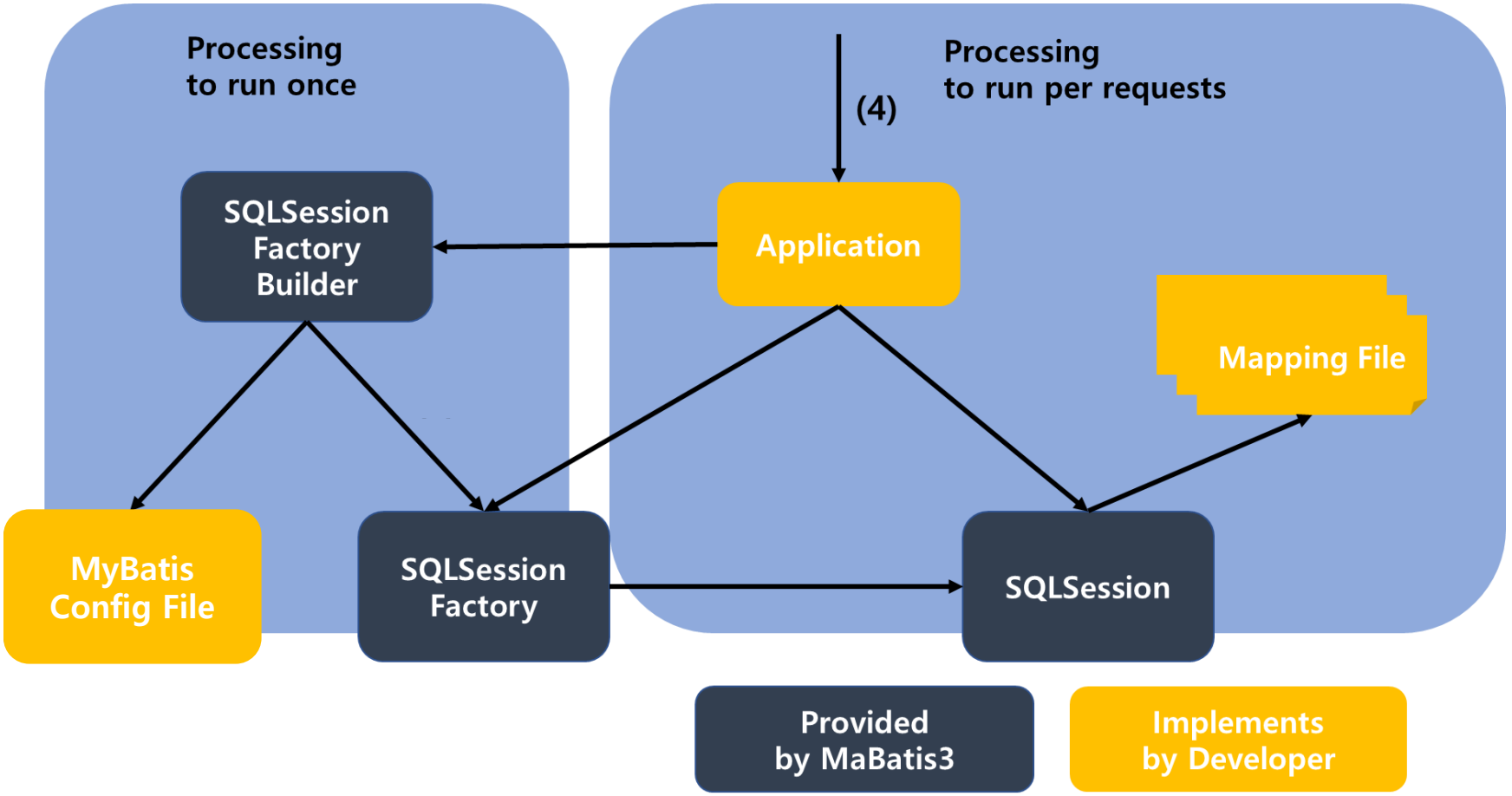
자동으로 and, or가 앞의 조건을 만족하지 않을때는 사라진다.

set

구문형식	속성	사용법
<set>		<set> <if test="writer != null"> writer = #{writer}, </if> <if test="title != null"> title = #{title} </if> </set>

자동으로 , 가 뒤의 조건을 만족하지 않을때 사라진다.

MyBatis3의 주요 컴포넌트



https://velog.io/@se_bb/Spring-Framework-15%EA%B0%95-MyBaits-%EC%86%8C%EA%B0%9C

목적

Mybatis를 쓰는 목적
-> java 프로그래밍과 sql의 분리
-> db connect하는 작업을 매번 하는 것들 피함
-> 동적 sql 생성

우리가 작성해야 할 것들.

Configuration- mybatis-config.xml
Sql Mapper – mapper.xml

SqlSessionFactoryBuilder-> SqlSessionFactory-> SqlSession

SqlSession을 불러와서, mapper.xml에 들어있는 sql문을 선택해서 실행시킨다.

SqlSession 클래스 중요 메소드

이름	설명
java.util.List	selectList(String id)
java.util.List	selectList(String id, Object parameterObject)
java.lang.Object	selectOne(String id)
java.lang.Object	selectOne(String id, Object parameterObject)
int	insert(String id, Object parameter)
int	delete(String id, Object parameter)
int	update(String id, Object parameter)

Mybatis-config.xml – (1)

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <environments default="development">
   <environment id="development">
    <transactionManager type="JDBC"/>
    <dataSource type="POOLED">
     <property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
     <property name="url"  value="jdbc:oracle:thin:@localhost:1521:xe"/>
     <property name="username" value="c##heejin"/>
     <property name="password" value="heejin"/>
    </dataSource>
   </environment>
  </environments>

 <mappers>
  <mapper resource="db/oracle/mapper.xml"/>
 </mappers>
</configuration>
```

Mybatis-config.xml – (2)

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <properties resource="db.properties" />

  <environments default="development">
   <environment id="development">
    <transactionManager type="JDBC"/>
    <dataSource type="POOLED">
     <property name="driver" value="${driver}"/>
     <property name="url"  value="${url}"/>
     <property name="username" value="${username}"/>
     <property name="password" value="${password}"/>
    </dataSource>
   </environment>
  </environments>

 <mappers>
  <mapper resource="db/oracle/mapper.xml"/>
 </mappers>
</configuration>
```

db .properties

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:xe
username=c##heejin
password=heejin
```

DB에 table 생성 및 자료 넣기

```
create table mybatis_board(
    no number(5) primary key
    , title varchar2(100) not null
    , writer varchar2(100) not null
    , content varchar2(1000) not null
    , reg_date date default sysdate
);
create sequence seq_mybatis_board_no nocache;
drop table mybatis_board;
desc mybatis_board;

insert into mybatis_board(no, title, writer, content)
values
  (seq_mybatis_board_no.nextval, '제목3', '작가3', '내용3');

select no, title, writer, content, reg_date
from mybatis_board
order by no desc;
```

mapper.xml – insert

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="boardDAO">
  <insert id="insert" parameterType="kr.ac.kopo.BoardVO">
   insert into mybatis_board( no, title, writer, content)
     values (seq_mybatis_board_no.nextval, #{title},
               #{writer}, #{content})
  </insert>
</mapper>
```

1. parameterType은 package포함 주소를 작성한다.
2. Sql문에 ;을 넣지 않는다.
3. Insert문은 return 값이 없기 때문에, returnType을 설정해 주지 않는다.
4. sqlSession은 commit을 따로 해주어야 한다. 자동으로 안해줌

mapper.xml – select

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="boardDAO">

 <select id="selectAll" resultType="kr.ac.kopo.BoardVO">
  select no, title, writer, content, reg_date
  from mybatis_board
  order by no desc
 </select>

 <select id="selectAll2" resultType="kr.ac.kopo.BoardVO">
  select no, title, writer, content,
        to_char(reg_date, 'yyyy-mm-dd') as regDate
  from mybatis_board
  order by no desc
 </select>
</mapper>
```

1. select문은 return 있으므로 returnType으로 설정한다.

mapper.xml – update

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="boardDAO">

 <update id="updateOne" parameterType="kr.ac.kopo.BoardVO">
       update mybatis_board
       set title=#{title}
       where no=#{no}
 </update>

</mapper>
```

mapper.xml – delete

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="boardDAO">

        <delete id="deleteOne" parameterType="int">
          delete from mybatis_board
          where no = #{no}
        </delete>

</mapper>
```

간결한 활용1 - <typeAlias>

1. 긴 주소를 typeAlias 를 사용하여 짧게 줄여 사용한다.

Mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>

  <typeAliases>
    <typeAlias type="kr.ac.kopo.BoardVO" alias="boardVO"/>
  </typeAliases>
```

간결한 활용2 – <sql>, <include>

1. 자주 쓰이는 sql문을 한번만 선언하고, 가져다 쓴다.
   <sql>, <include>를 활용한다

Mapper.xml

```
<mapper namespace="boardDAO">
  <sql id="selectcommon" >
    select no, title, writer, content,
           to_char(reg_date, 'yyyy-mm-dd') as regDate
    from mybatis_board
  </sql>

  <select id="selectAll3" resultType="kr.ac.kopo.BoardVO">
       <include refid="selectcommon"/>
       order by no desc
  </select>
```

간결한 활용3 – ResultMap

1. DB의 column과 Bean의 변수가 다를경우 활용

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"https://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="boardDAO">
    <resultMap type="kr.ac.kopo.BoardVO" id="boardMap">
        <result property="no" column="no"/>
        <result property="title" column="title"/>
        <result property="writer" column="writer"/>
        <result property="content" column="content"/>
        <result property="regDate" column="reg_date"/>
    </resultMap>
    <select id="selectAll4" resultMap="boardMap">
        select no, title, writer, content,
                to_char(reg_date, 'yyyy-mm-dd') as regDate
         from mybatis_board
        order by no desc
    </select>
</mapper>
```

Dynamic SQL - where

```
<select id="selectWhere" parameterType="boardVO" resultMap="boardMap">
  select no, title, writer, content,
      to_char(reg_date, 'yyyy-mm-dd') as regDate
    from mybatis_board
  <where>
     <if test="title!=null" > title = #{title} </if>
     <if test="writer!=null"> and writer = #{writer} </if>
  </where>
</select>
```

```
BoardVO board1 = new BoardVO();
board1.setTitle("제목3");  //삭제 가능
board1.setWriter("작가3");  //삭제가능
List<BoardVO> boardlist =
  sqlSession.selectList("boardDAO.selectWhere", board1);
```

Dynamic SQL - foreach

```
<select id="selectNumbers" parameterType="kr.ac.kopo.IntListVO" resultMap="boardMap">
   select no, title, writer, content, to_char(reg_date, 'yyyy-mm-dd') as regDate
     from mybatis_board
    where no in
      <foreach collection="numbers" item="number" open="(" close=")" separator=",">
          #{number}
      </foreach>
</select>
```

```
<select id="selectNumbers2" parameterType="int[]" resultMap="boardMap">
  select no, title, writer, content, to_char(reg_date, 'yyyy-mm-dd') as regDate
    from mybatis_board
   where no in
         <foreach collection="array" item="number" open="(" close=")" separator=",">
           #{number}
         </foreach>
</select>
```

```
<select id="selectNumbers3" parameterType="int[]" resultMap="boardMap">
  select no, title, writer, content, to_char(reg_date, 'yyyy-mm-dd') as regDate
   from mybatis_board
  where no in
         <foreach collection="array" index="index" open="(" close=")" separator=",">
          ${array[index]}
         </foreach>
</select>
```

Map으로 결과 저장하기

```
<select id="selectMap2" parameterType="int" resultType="java.util.Map">
select no, title, writer, content, to_char(reg_date, 'yyyy-mm-dd') as regDate
 from mybatis_board
  where no=#{no}
</select>
<select id="selectMap" parameterType="java.util.Map" resultMap="boardMap">
select no, title, writer, content, to_char(reg_date, 'yyyy-mm-dd') as regDate
 from mybatis_board
  where title = #{title} and writer=#{writer}
</select>
```

```
public void selectMap(){//input을 map으로
          Map<String, String> map = new HashMap();
          map.put("title", "제목3");
          map.put("writer", "작가3");
          List<BoardVO> boardlist = sqlSession.selectList("boardDAO.selectMap", map);
          for (BoardVO boardVO : boardlist) {
                    System.out.println(boardVO.toString());
          }
}
public void selectMap2(){//output을 map으로
          Map<String, Object> resultmap= sqlSession.selectOne("boardDAO.selectMap2",2);
          Set<String> keys = resultmap.keySet();
          for (String key: keys) {
                    System.out.println(key + " : " + resultmap.get(key));
          }
}
```

Spring db

datasource 설정

✓ DriverManager를 이용한 DataSource 설정

: 일반 JDBC 방식

```
<bean id="dataSource"
      class="org.springframework.jdbc.datasource.
                                    DriverManagerDataSource"
      p:driverClassName="oracle.jdbc.driver.OracleDriver"
      p:url="jdbc:oracle:thin:@localhost:1521:XE"
      p:username="hr"
      p:password="hr" />
```

datasource 설정

✓ JNDI를 이용한 DataSource 설정

: 웹서버(WebLogic, Tomcat등)에 등록된 JNDI로부터 설정

```
<jee:jndi-lookup id="dataSource"
                 jndi-name="java:comp/env/jdbc/bitdb"/>
```

datasource 설정

✓ 커넥션 풀을 이용한 DataSource 설정

: DBCP(Apache Commons Database Connection Pool) API를 이용한 설정

1. **Apache Commons DBCP**

org.apache.commons » commons-dbcp2

Apache Commons DBCP software implements Database Connection Pooling

Last Release on Aug 4, 2021

```
<bean id="datasource" class="org.apache.commons.dbcp2.BasicDataSource">
  <property name="driverClassName"
            value="oracle.jdbc.driver.OracleDriver"/>
  <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
  <property name="username" value="userid"/>
  <property name="password" value="userpassword"/>
</bean>
```

Spring에서 Properties를 활용 하여 접속

```
<!-- config/spring/spring-config.xml -->

<context:property-placeholder
  location="classpath:config/db/db.properties"/>


<bean id="dataSource"
            class="org.apache.commons.dbcp2.BasicDataSource">
  <property name="driverClassName" value="${jdbc.driver}"/>
  <property name="url" value="${jdbc.url}"/>
  <property name="username" value="{jdbc.username}"/>
  <property name="password" value="{jdbc.password}"/>
</bean>
```

Spring junit

Library

- Junit
- spring-test

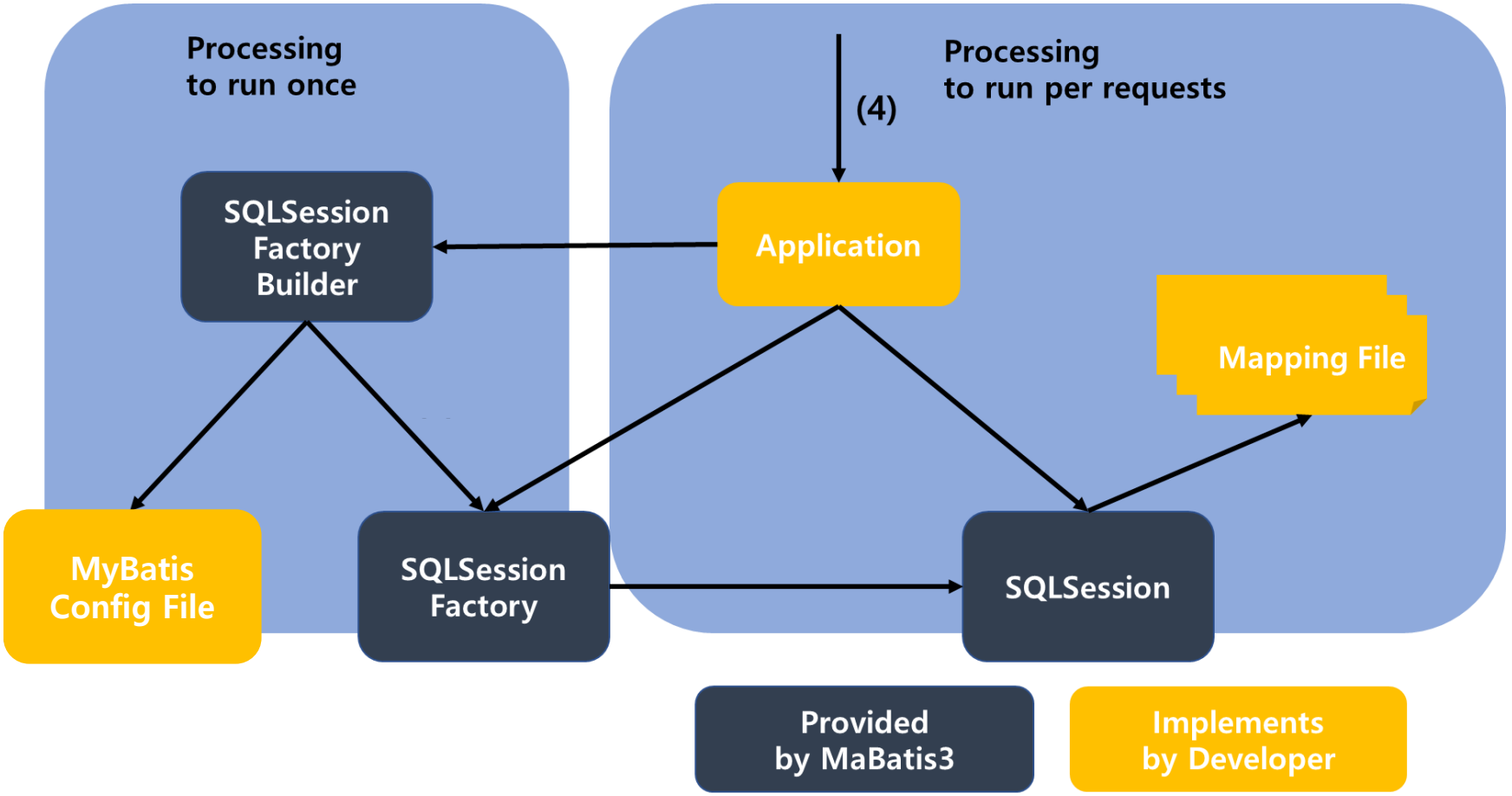
Spring - Junit

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration
          (locations = {"classpath:config/spring/spring-config.xml"})
public class DSTest {
        @Autowired
        private DataSource dataSource;

        @Test
        public void dsTest() {
                assertNotNull(dataSource);
        }
}
```

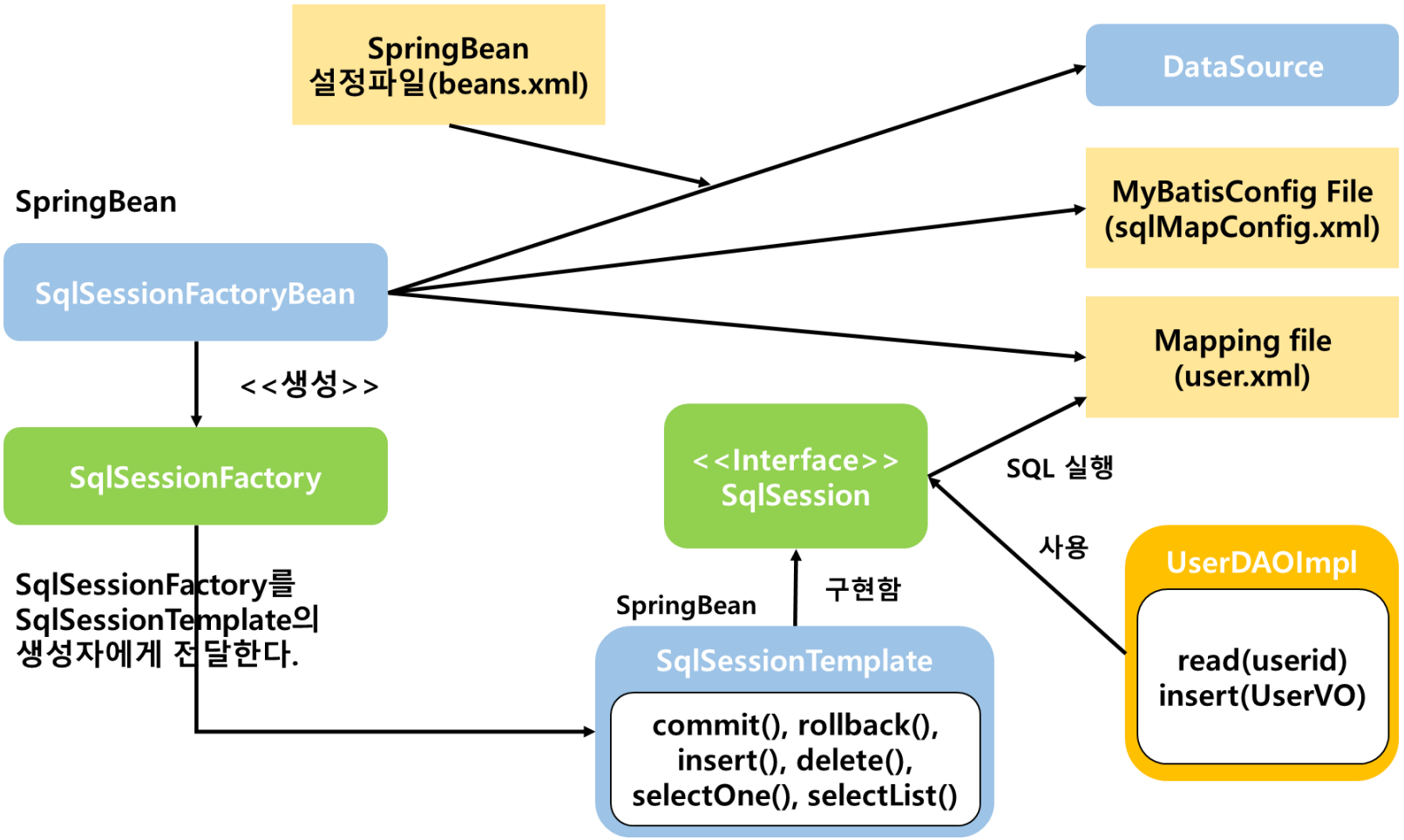
Src/test/java 폴더 아래에 작성한다.

MyBatis3의 주요 컴포넌트



https://velog.io/@se_bb/Spring-Framework-15%EA%B0%95-MyBaits-%EC%86%8C%EA%B0%9C

MaBatis-Spring의 주요 컴포넌트



https://velog.io/@se_bb/Spring-Framework-15%EA%B0%95-MyBaits-%EC%86%8C%EA%B0%9C

spring + mybatis 연동 - xml

- ✓ mybatis-spring-x.x.x.jar, mybatis-x.x.x.jar 파일을 클래스 패스 설정
- ✓ DAO에서 SqlSession을 얻기 위해 XML 문서에 *SqlSessionTemplate* *를 선언*

```
<bean id="dataSource" destroy-method="close"
      class="org.apache.commons.dbcp.BasicDataSource">
   ..... 생략
</bean>

<bean id="sqlSessionFactory"
      class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="configLocation"
            value="classpath:config/mybatis/mybatis-config.xml" />
</bean>

<bean id="sqlSessionTemplate"
      class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg ref="sqlSessionFactory" />
</bean>
```

mybatis-config.xml

```
<configuration>
    <typeAliases>
        <typeAlias alias="xxx" type="xxx.vo.XxxVO" />
    </typeAliases>
    <mappers>
        <mapper resource="config/sqlmap/oracle/xxx.xml" />
    </mappers>
</configuration>
```

Spring transactions

트랜잭션

- ✓ Container 가 제공되는 가장 대표적 서비스
- ✓ 설정파일내 TransactionManager 설정만으로 소스코드 내에 Transaction 관련 코드를 사용하지 않고 자동 Transaction 이 가능
- ✓ Spring 에서는 서로 다른 트랜잭션 API를 지원하며 지원 종류는 JDBC, JTA, JDO, JPA, 하이버네이트 등이 있음
- ✓ 트랜잭션 사용방법은 환경설정 파일인 xml 에 선언하여 사용하는 방법과 프로그램에서 직접 제어하는 방법 2가지가 있음

AOP 방식의 트랜잭션

1. <beans> 태그에 xmlns:tx 관련 부분을 추가

```
<beans xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="“http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd">
```

2. DataSource 추가

```
<bean id="dataSource" 생략…  />
```

3. 트랜잭션 매니져 설정 : 실제 트랜잭션 관리 처리

```
<bean id="transactionManager"
      class=
   "org.springframework.jdbc.datasource.DataSourceTransactionManager"
      p:dataSource-ref="dataSource" />
```

AOP 방식의 트랜잭션

4. 트랜잭션 매니져를 어드바이스로 설정

```
<tx:advice id="txAdvice"
        transaction-manager="transactionManager">
     <tx:attributes>
        <tx:method name="tran*" />
        <tx:method name="select*" read-only="true" />
    </tx:attributes>
</tx:advice>
```

5. 트랜잭션 AOP 설정을 통한 적용

```
<aop:config>
    <aop:pointcut id="tranMethod"
          expression="execution(public * member.*Service.*(..))“/>
    <aop:advisor advice-ref="txAdvice“ pointcut-ref="tranMethod" />
</aop:config>
```

어노테이션 방식의 트랜잭션

1. annotation-driven 설정

```
<tx:annotation-driven
            transaction-manager="transactionManager" />
```

2. 메소드에 어노테이션 표기법 추가

```
@Service("memberService")
public class MemberServiceImpl implements MemberService {
    …… 생략
    @Transactional
    public void registMember(MemberVO memberVO) throws Exception {
        memberDAO.insertMember(memberVO);
    }
}
```