

DAY - 05

Spring, MyBatis 연동

- Mybatis 특징

- Apache의 Ibatis 프레임워크가 Data Mapper 프레임워크로 탄생함.
- 2010년에 Ibatis가 Apache에서 탈퇴하여 Google로 흡수되면서 이름을 Mybatis로 변경함.
- Mybatis는 Ibatis로부터 파생되었기 때문에 기본 개념과 문법은 Ibatis와 거의 유사함.
- Mybatis는 **SQL 명령어를 외부의 XML 파일에 분리시키고 DB 연동을 1줄의 자바 코드로 처리**하는 프레임워크임.

MyBatis Quick Start

- Dependency 설정 (pom.xml)

```
<!-- Mybatis -->
```

```
<dependency>
```

```
    <groupId>org.mybatis</groupId>
```

```
    <artifactId>mybatis</artifactId>
```

```
    <version>3.3.1</version>
```

```
</dependency>
```

```
<!-- Ibatis -->
```

```
<dependency>
```

```
    <groupId>org.apache.ibatis</groupId>
```

```
    <artifactId>ibatis-core</artifactId>
```

```
    <version>3.0</version>
```

```
</dependency>
```

▼ Maven Dependencies

> h2-1.4.191.jar - C:\Users\GURUM\m2\repository\com\h2\h2-1.4.191.jar

> mybatis-3.3.1.jar - C:\Users\GURUM\m2\repository\org\mybatis\mybatis-3.3.1.jar

> ibatis-core-3.0.jar - C:\Users\GURUM\m2\repository\org\apache\ibatis\ibatis-core-3.0.jar

> spring-context-4.2.4.RELEASE.jar - C:\Users\GURUM\m2\repository\org\springframework\spring-context-4.2.4.RELEASE.jar

> spring-aop-4.2.4.RELEASE.jar - C:\Users\GURUM\m2\repository\org\springframework\spring-aop-4.2.4.RELEASE.jar

> aopalliance-1.0.jar - C:\Users\GURUM\m2\repository\org\springframework\spring-aop-4.2.4.RELEASE.jar

> spring-beans-4.2.4.RELEASE.jar - C:\Users\GURUM\m2\repository\org\springframework\spring-beans-4.2.4.RELEASE.jar

- VO 클래스 작성 (BoardVO.java)

```
import java.util.Date;
```

```
public class BoardVO {  
    private int seq;  
    private String title;  
    private String writer;  
    private String content;  
    private Date regDate;  
    private int cnt;  
    private String searchCondition;  
    private String searchKeyword;  
  
    // Getter / Setter  
}
```

- SQL Mapper 작성 (board-mapping.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="BoardDAO">
```

```
    <insert id="insertBoard">
```

```
        insert into board(seq, title, writer, content)
```

```
        values((select nvl(max(seq), 0)+1 from board),#{title},#{writer},#{content})
```

```
    </insert>
```

```
    <select id="getBoardList" resultType="board">
```

```
        select * from board
```

```
        where title like '%'||#{searchKeyword}||'%'
```

```
        order by seq desc
```

```
    </select>
```

```
</mapper>
```

- MyBatis 메인 환경설정 파일 (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>
```

```
<!-- Properties 파일 설정 -->
```

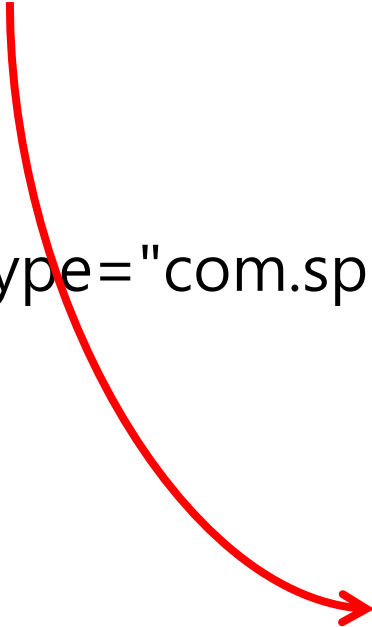
```
<properties resource="db.properties"/>
```

```
<!-- Alias 설정 -->
```

```
<typeAliases>
```

```
<typeAlias alias="board" type="com.springbook.biz.board.BoardVO"/>
```

```
</typeAliases>
```



```
jdbc.driverClassName=org.h2.Driver  
jdbc.url=jdbc:h2:tcp://localhost/~ /test  
jdbc.username=sa  
jdbc.password=
```

➔ 계속

<!-- DataSource 설정 -->

<environments default="development">

<environment id="development">

<transactionManager type="JDBC"/>

<dataSource type="POOLED">

<property name="driver" value="\${jdbc.driverClassName}"/>

<property name="url" value="\${jdbc.url}" />

<property name="username" value="\${jdbc.username}"/>

<property name="password" value="\${jdbc.password}"/>

</dataSource>

</environment>

</environments>

<!-- Sql Mapper 설정 -->

<mappers>

<mapper resource="mappings/board-mapping.xml"/>

</mappers>

</configuration>

- SessionFactoryBean 클래스

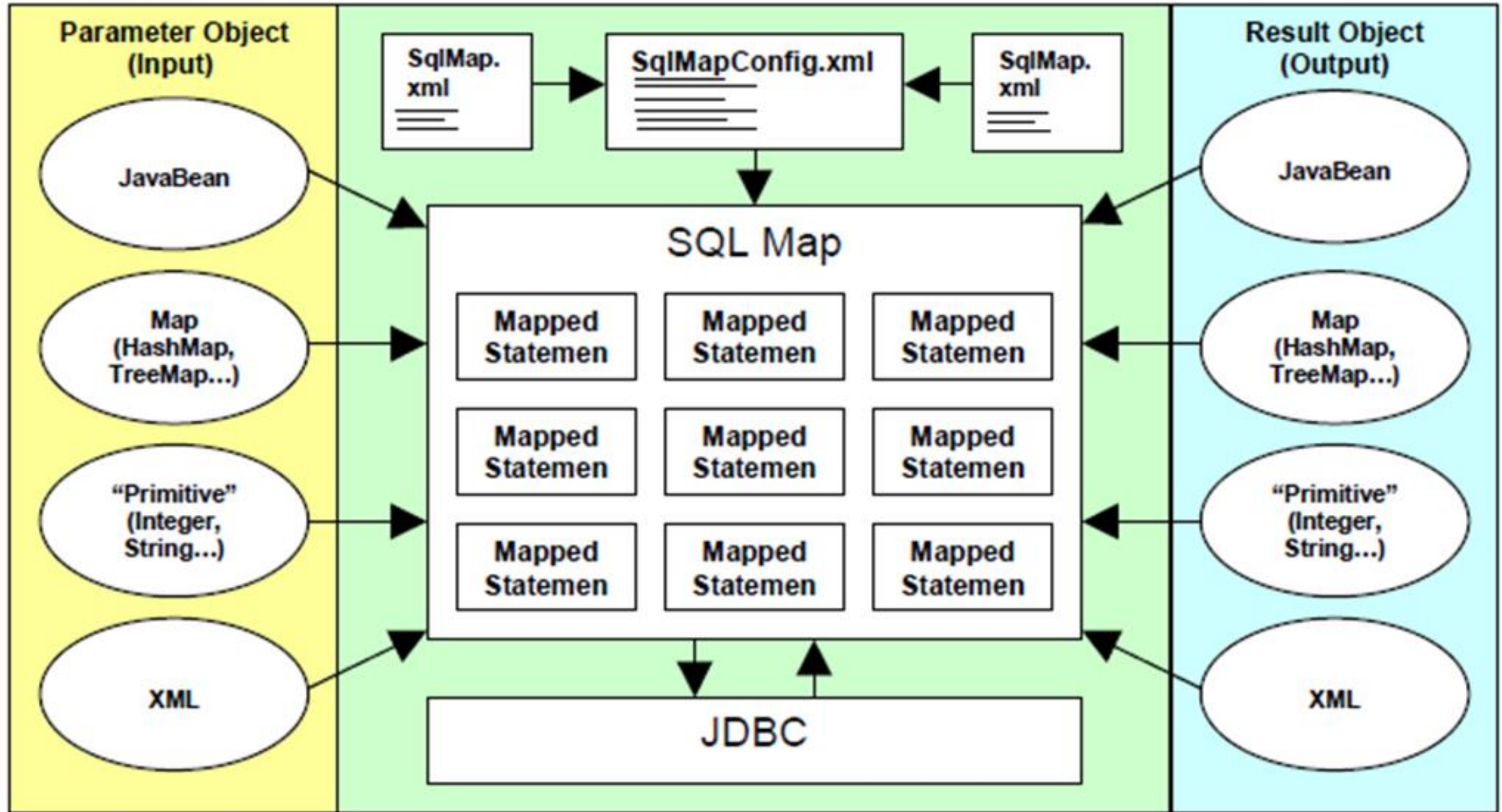
```
public class SqlSessionFactoryBean {  
    private static SqlSessionFactory sessionFactory = null;  
    static {  
        try {  
            if (sessionFactory == null) {  
                Reader reader = Resources.getResourceAsReader("sql-map-config.xml");  
                sessionFactory = new SqlSessionFactoryBuilder().build(reader);  
            }  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
  
    public static SqlSession getSqlSessionInstance() {  
        return sessionFactory.openSession();  
    }  
}
```

- DAO 클래스 작성 (BoardDAO.java)

```
public class BoardDAO {  
    private SqlSession mybatis;  
  
    public BoardDAO() {  
        mybatis = SqlSessionFactoryBean.getSqlSessionInstance();  
    }  
  
    public void insertBoard(BoardVO vo) {  
        mybatis.insert("BoardDAO.insertBoard", vo);  
        mybatis.commit();  
    }  
  
    public List<BoardVO> getBoardList(BoardVO vo) {  
        return mybatis.selectList("BoardDAO.getBoardList", vo);  
    }  
}
```

Mapper XML 설정

- MyBatis 구조



- Mapper XML 구조

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

DTD 선언

```
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
```

```
<mapper namespace="...">
```

SQL Mapping

```
<insert id="insertBoard"></insert>  
<update id="updateBoard"></update>  
<delete id="deleteBoard"></delete>  
<select id="getBoard"></select>  
<select id="getBoardList"></select>
```

```
</mapper>
```

- id 속성과 namespace

- namespace는 유일한 id를 생성할 수 있도록 한다.

board-mapping.xml	user-mapping.xml
<pre><mapper namespace="BoardDAO"> <select id="getTotalCount" resultType="int"> select count(*) from board </select> </mapper></pre>	<pre><mapper namespace="UserDAO"> <select id="getTotalCount" resultType="int"> select count(*) from users </select> </mapper></pre>

- parameterType과 resultType 속성

sql-map-config.xml

```
<typeAliases>
    <typeAlias alias="board" type="com.springbook.biz.board.BoardVO"/>
</typeAliases>
```

board-mapping.xml

```
<select id="getBoard" parameterType="board"
        resultType="board">
    select * from board where seq = #{seq}
</select>
```

- resultMap 속성

```
<mapper namespace="Board">
```

```
  <resultMap id="boardResult" type="board">
```

```
    <id property="seq" column="SEQ" />
```

```
    <result property="title" column="TITLE" />
```

```
    <result property="writer" column="WRITER" />
```

```
    <result property="content" column="CONTENT" />
```

```
    <result property="regDate" column="REGDATE" />
```

```
    <result property="cnt" column="CNT" />
```

```
  </resultMap>
```

```
  <select id="getBoardList" parameterType="board" resultMap="boardResult">
```

```
    select * from board
```

```
    where title like '%'||#{searchKeyword}||'%'
```

```
    order by seq desc
```

```
  </select>
```

```
</mapper>
```


- SqlSession 메소드

- **selectOne()** : 데이터 하나를 조회할 때 사용. (쿼리 결과가 두 개 이상일 때 예외 발생)
 - public Object selectOne(String statement)
 - public Object selectOne(String statement, Object parameter)
- **selectList()** : 여러 데이터 목록을 조회할 때 사용.
 - public List selectList(String statement)
 - public List selectList(String statement, Object parameter)
- **insert(), update(), delete()** : INSERT, UPDATE, DELETE SQL 실행할 때 사용.
 - public int insert(String statement, Object parameter)
 - public int update (String statementId, Object parameterObject) throws SQLException
 - public int delete (String statementId, Object parameterObject) throws SQLException

Spring – MyBatis 연동

- 라이브러리 추가 (pom.xml)

```
<!-- Mybatis -->
```

```
<dependency>
```

```
    <groupId>org.mybatis</groupId>
```

```
    <artifactId>mybatis</artifactId>
```

```
    <version>3.3.1</version>
```

```
</dependency>
```

```
<!-- Mybatis Spring -->
```

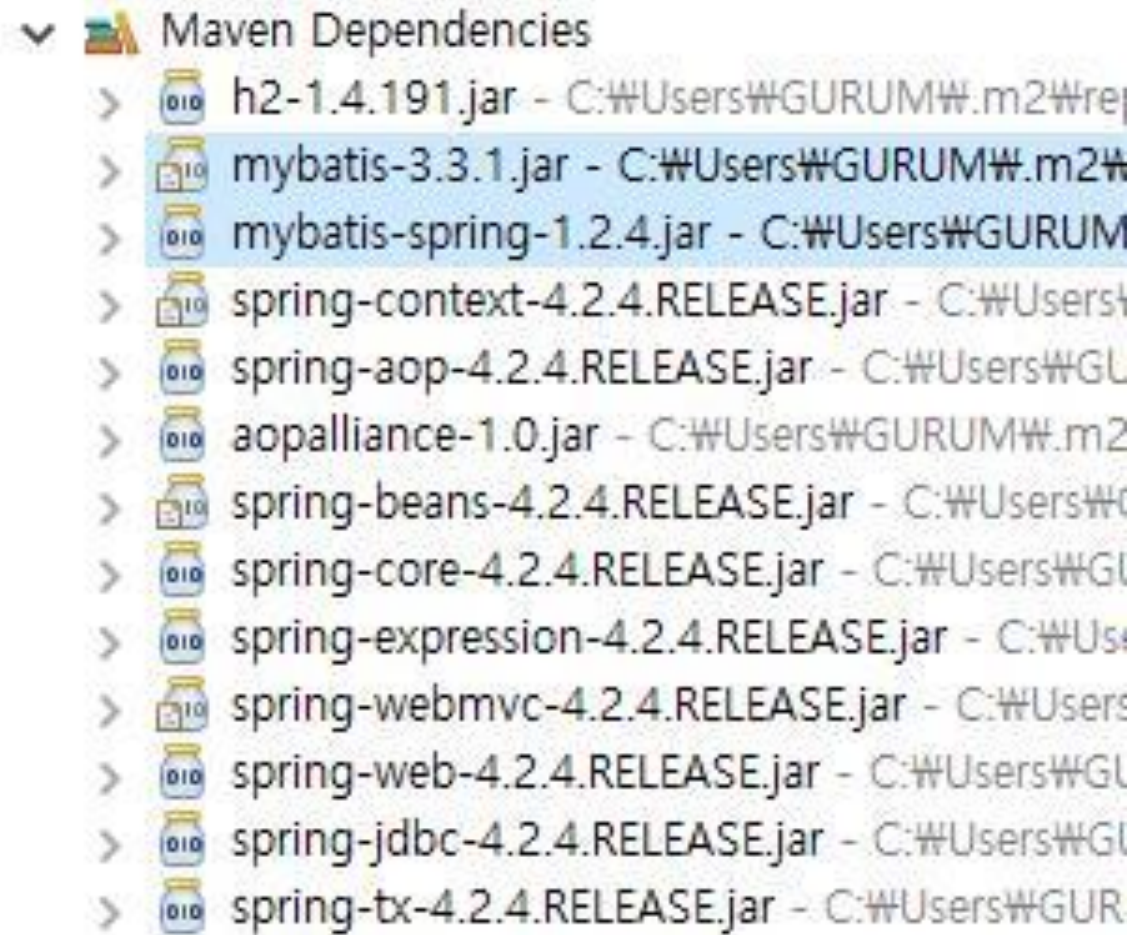
```
<dependency>
```

```
    <groupId>org.mybatis</groupId>
```

```
    <artifactId>mybatis-spring</artifactId>
```

```
    <version>1.2.4</version>
```

```
</dependency>
```



- MyBatis 메인 설정파일 수정

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>
  <!-- Alias 설정 -->
  <typeAliases>
    <typeAlias alias="board" type="com.springbook.biz.board.BoardVO"/>
  </typeAliases>

  <!-- Sql Mapper 설정 -->
  <mappers>
    <mapper resource="mappings/board-mapping.xml"/>
  </mappers>
</configuration>
```

- Spring 설정 파일 수정 (applicationContext.xml)

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

- DAO 클래스 (BoardDAOMyBatis.java)

@Repository

```
public class BoardDAOMybatis extends SqlSessionDaoSupport {
```

@Autowired

```
public void setSqlSessionFactory(SqlSessionFactory sqlSessionFactory) {  
    super.setSqlSessionFactory(sqlSessionFactory);  
}
```

```
public void insertBoard(BoardVO vo) {  
    System.out.println("===> Mybatis로 insertBoard() 기능 처리");  
    getSqlSession().insert("BoardDAO.insertBoard", vo);  
}
```

```
public void updateBoard(BoardVO vo) {  
    System.out.println("===> Mybatis로 updateBoard() 기능 처리");  
    getSqlSession().update("BoardDAO.updateBoard", vo);  
}
```

➔ 계속

- DAO 클래스 (BoardDAOMyBatis.java)

```
public void deleteBoard(BoardVO vo) {  
    System.out.println("===> Mybatis로 deleteBoard() 기능 처리");  
    getSqlSession().delete("BoardDAO.deleteBoard", vo);  
}  
  
public BoardVO getBoard(BoardVO vo) {  
    System.out.println("===> Mybatis로 getBoard() 기능 처리");  
    return (BoardVO) getSqlSession().selectOne("BoardDAO.getBoard", vo);  
}  
  
public List<BoardVO> getBoardList(BoardVO vo) {  
    System.out.println("===> Mybatis로 getBoardList() 기능 처리");  
    return getSqlSession().selectList("BoardDAO.getBoardList", vo);  
}  
}
```