Mybatis

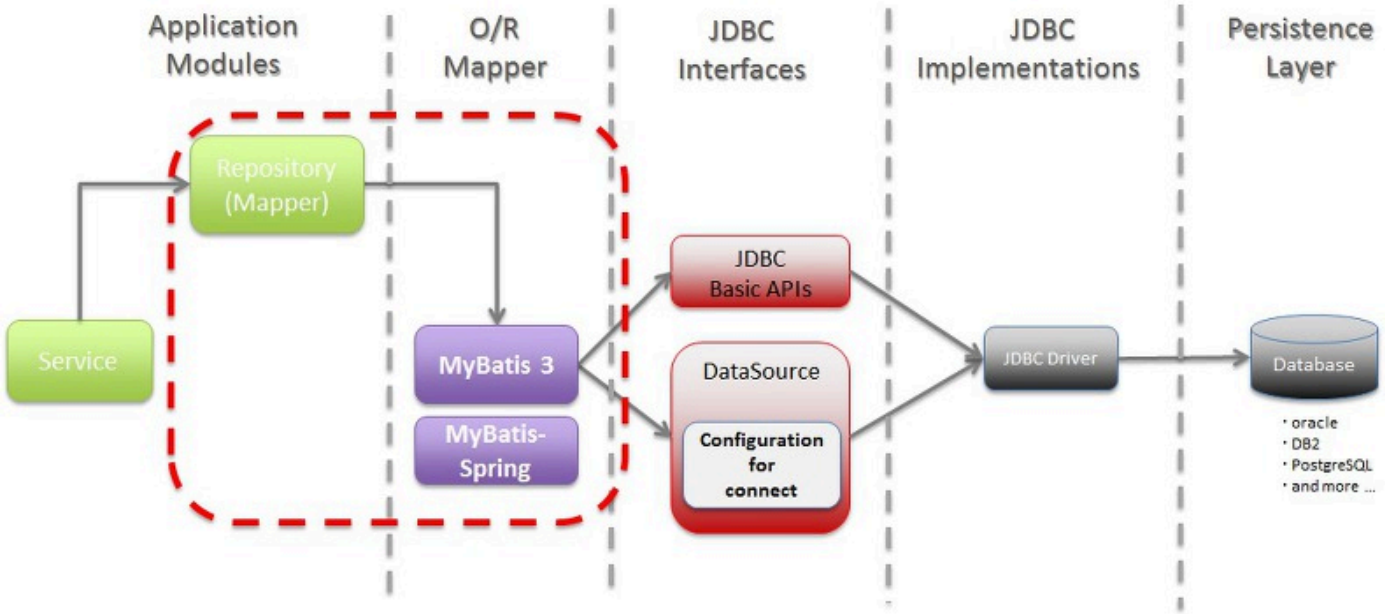
MyBatis 란..??

- Java Object와 SQL문 사이에서 자동 Mapping 기능을 지원하는 ORM(Object Relational Mapping) Framework

MyBatis의 특징

- JDBC를 통해 RDBMS(관계형 데이터베이스 관리 시스템)에 엑세스하는 작업을 캡슐화하고 기존 JDBC의 중복작업을 간소화
- XML파일의 형태인 mapper를 통해 프로그램 코드로 부터 SQL 쿼리를 분리
- 쿼리 실행 결과를 Java 객체와 자동으로 매핑
- 빠른 개발과 생산성의 향상
- 기존 JDBC보다 사용하기 편리
- 학습의 부담이 적음

Mybatis의 아키텍처



기존 JDBC 프로그래밍의 경우 Repository(DAO)에서 곧바로 JDBC API쪽으로 접근하여 DB를 연결하였지만, 위의 그림에 나와있듯이 Mybatis을 사용할 경우 Repository와 JDBC API사이에 MyBatis가 위치함으로써 편리한 Access를 제공한다.

Mybatis의 Mapper(매퍼)

DAO클래스를 사용하면

- SQL문만 수정하더라도 클래스를 다시 컴파일 해야한다.
- SQL문만 별도로 한 곳에 모아서 관리하기 어렵다.

마이바티스는 Mapper를 통해 SQL문을 Java코드로 부터 완전히 분리한다.

따라서 SQL 문에 대한 통합 관리를 할 수 있다.

매퍼는 xml 문서이므로 다음과 같은 포맷을 가진다.

```
<?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="매퍼를 호출하기 위한 인터페이스">
    <select id="호출할메서드명1">
        select 쿼리문
    </select>
    <insert id="호출할메서드명2">
        insert 쿼리문
    </insert>
    <update id="호출할메서드명3">
        update 쿼리문
    </update>
    <delete id="호출할메서드명4">
        delete 쿼리문
    </delete>
</mapper>
```

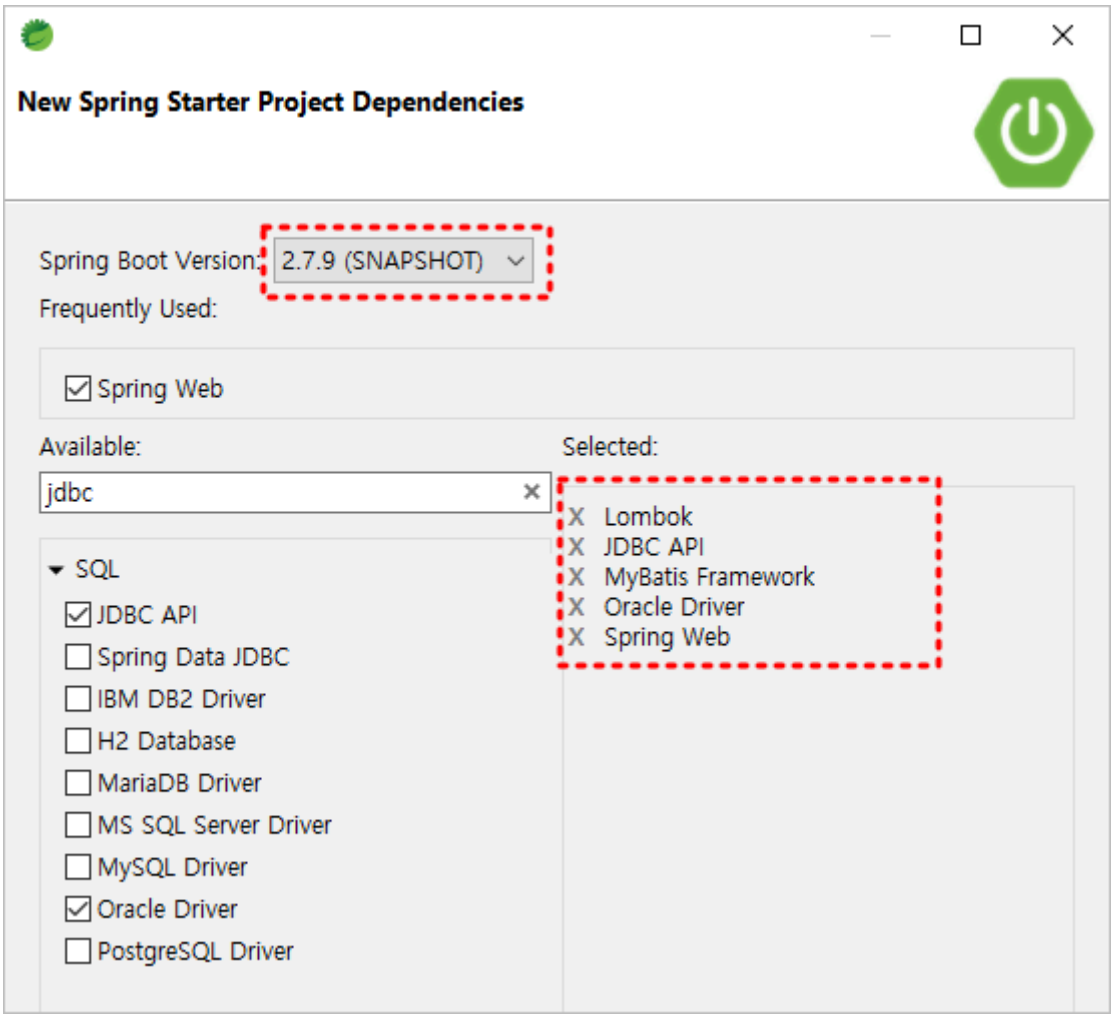
Mapper 의 Doctype

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

프로젝트명 : B07aMybatis

준비사항

- 의존설정 : Spring Web, Lombok, JDBC API, Oracle Driver, Mybatis Framework
- JSP 사용을 위한 설정을 한다.
- Refresh Gradle Project 를 눌러 적용한다.



프로퍼티 : resources/application.properties

```
# 포트 설정
server.port=8586
# JSP 설정
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp

# oracle set
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver
spring.datasource.url=jdbc:oracle:thin:@localhost:1521:xe
spring.datasource.username=musthave
spring.datasource.password=1234
```

```
12
13 # mybatis
14 mybatis.mapper-locations=classpath:mybatis/mapper/**/**.xml
15
```

서비스 인터페이스 : com.edu.springboot.jdbc.IMemberService.java

```java
package com.edu.springboot.jdbc;

import java.util.List;

@Mapper
public interface IMemberService {

    public List<MemberDTO> select();
    public int insert(MemberDTO memberDTO);
    public MemberDTO selectOne(MemberDTO memberDTO);
    public int update(MemberDTO memberDTO);
    public int delete(MemberDTO memberDTO);
}
```

DTO : com.edu.springboot.jdbc.MemberDTO.java

```java
package com.edu.springboot.jdbc;

import lombok.Data;

@Data
public class MemberDTO {
    private String id;
    private String pass;
    private String name;
    private String regidate;
}
```

매퍼 : resources/mybatis/mapper/MemberDAO.xml [위 설명부분 참조]

```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.edu.springboot.jdbc.IMemberService">

</mapper>

```

여기까지 작성하세요.

회원목록

home : webapp/WEB-INF/views/home.jsp

```jsp
<body>
    <h2>스프링 부트 프로젝트</h2>
    <ul>
        <li><a href="/">루트</a></li>
    </ul>

    <h2>Mybatis로 구현한 회원관리</h2>
    <ul>
        <li><a href="./list.do">회원목록</a></li>
    </ul>
</body>
```

컨트롤러 : com.edu.springboot.MainController.java

```java
package com.edu.springboot;

import org.springframework.beans.factory.annotation.Autowired;

@Controller
public class MainController {

    @Autowired
    IMemberService dao;

    @RequestMapping("/")
    public String home() {
        return "home";
    }

    //회원목록
    @RequestMapping("/list.do")
    public String member2(Model model) {
        model.addAttribute("memberList", dao.select());
        return "list";
    }

```

여기까지 작성하세요.

매퍼 : resources/mybatis/mapper/MemberDAO.xml

```xml
    <select id="select"
        parameterType="com.edu.springboot.jdbc.MemberDTO"
        resultType="com.edu.springboot.jdbc.MemberDTO">
        select * from member order by regidate desc
    </select>

```

뷰 : webapp/WEB-INF/views/list.jsp

```
<body>
    <h2>회원리스트</h2>
    <table border="1">
        <tr>
            <th>아이디</th>
            <th>패스워드</th>
            <th>이름</th>
            <th>가입일</th>
            <th></th>
        </tr>
        <c:forEach items="${memberList }" var="row" varStatus="loop">
        <tr>
            <td>${row.id }</td>
            <td>${row.pass }</td>
            <td>${row.name }</td>
            <td>${row.regidate }</td>
            <td>
                <a href="edit.do?id=${row.id }">수정</a>
                <a href="delete.do?id=${row.id }">삭제</a>
            </td>
        </tr>
        </c:forEach>
    </table>
    <a href="regist.do">회원등록</a>
</body>
```

여기까지 작성하세요.

등록하기

컨트롤러 : com.edu.springboot.MainController.java

```
30      //회원등록
31      @RequestMapping(value="/regist.do", method=RequestMethod.GET)
32      public String member1() {
33          return "regist";
34      }
35      @RequestMapping(value="/regist.do", method=RequestMethod.POST)
36      public String member6(MemberDTO memberDTO) {
37          int result = dao.insert(memberDTO);
38          if(result==1) System.out.println("입력되었습니다.");
39          return "redirect:list.do";
40      }
41
```

매퍼 : resources/mybatis/mapper/MemberDAO.xml

```
15      <insert id="insert"
16          parameterType="com.edu.springboot.jdbc.MemberDTO">
17          insert into member (id, pass, name) values
18              (#{id}, #{pass}, #{name})
19      </insert>
```

뷰 : webapp/WEB-INF/views/regist.jsp

```
10  <body>
11      <h2>회원등록</h2>
12      <form action="regist.do" method="post">
13      <table border="1">
14          <tr>
15              <th>아이디</th>
16              <td><input type="text" name="id" value="" /></td>
17          </tr>
18          <tr>
19              <th>패스워드</th>
20              <td><input type="text" name="pass" value="" /></td>
21          </tr>
```

```
22          <tr>
23              <th>이름</th>
24              <td><input type="text" name="name" value="" /></td>
25          </tr>
26      </table>
27      <input type="submit" value="전송하기" />
28      </form>
29  </body>
```

여기까지 작성하세요.

수정하기

컨트롤러 : com.edu.springboot.MainController.java

```
42      //회원수정
43      @RequestMapping(value="/edit.do", method=RequestMethod.GET)
44      public String member3(MemberDTO memberDTO, Model model) {
45          memberDTO = dao.selectOne(memberDTO);
46          model.addAttribute("dto", memberDTO);
47          return "edit";
48      }
49      @RequestMapping(value="/edit.do", method=RequestMethod.POST)
50      public String member7(MemberDTO memberDTO) {
51          int result = dao.update(memberDTO);
52          if(result==1) System.out.println("수정되었습니다.");
53          return "redirect:list.do";
54      }
55
```

매퍼 : resources/mybatis/mapper/MemberDAO.xml

```
21      <select id="selectOne"
22          parameterType="com.edu.springboot.jdbc.MemberDTO"
23          resultType="com.edu.springboot.jdbc.MemberDTO">
24          select * from member where id=#{id}
25      </select>
26
27      <update id="update"
28          parameterType="com.edu.springboot.jdbc.MemberDTO">
29          update member set pass=#{pass},
30              name=#{name} where id=#{id}
31      </update>
32
```

뷰 : webapp/WEB-INF/views/edit.jsp

```
10  <body>
11      <h2>회원수정</h2>
12      <form action="edit.do" method="post">
13      <table border="1">
14          <tr>
15              <th>아이디(수정불가)</th>
16              <td><input type="text" name="id" value="${dto.id }"
17                  readonly/></td>
18          </tr>
19          <tr>
20              <th>패스워드</th>
21              <td><input type="text" name="pass" value="${dto.pass }" /></td>
22          </tr>
23          <tr>
24              <th>이름</th>
25              <td><input type="text" name="name" value="${dto.name }" /></td>
26          </tr>
27      </table>
28      <input type="submit" value="전송하기" />
29      </form>
30  </body>
```

여기까지 작성하세요.

삭제하기

컨트롤러 : com.edu.springboot.MainController.java

```
56      //회원삭제
57      @RequestMapping("/delete.do")
58      public String member4(MemberDTO memberDTO) {
59          int result = dao.delete(memberDTO);
60          if(result==1) System.out.println("삭제되었습니다.");
61          return "redirect:list.do";
62      }
63  }
```

매퍼 : resources/mybatis/mapper/MemberDAO.xml

```
33      <delete id="delete"
34          parameterType="com.edu.springboot.jdbc.MemberDTO">
35          delete from member where id=#{id}
36      </delete>
37
38  </mapper>
```

퀴즈]

현재 삭제 버튼을 누르는 경우 즉시 삭제된다. Javascript의 confirm() 함수로 물어본 후 '확인'을 누를때만 삭제될 수 있도록 구현하시오. 단, 전송방식은 post로 처리한다.