210723 Mybatis

1교시

간단하게 Mybatis에 대해서 알아보자

기존 work05_Web 워크 스페이스를 실행하여 기존에 사용하였던 프로젝트를 이용한다.

- 다음에 배울 Spring 에 들어가게되면 Mybatis 설정 방법은 또 달라진다.
  하지만 mapper 파일과 configFile이 설정되어야 한다는건 변하지않는다.

1교시에는 Mybatis

https://github.com/mybatis/mybatis-3/releases/tag/mybatis-3.5.6

의 3.5.6버전.jar를 다운받았다!

work05_Web/jsp01_MyBoard 프로젝트를 복사하여

jsp01_MyBoard_Mybatis라는 이름으로 프로젝트를 생성해주었다.

jsp01_MyBoard_Mybatis/WebContent/WEB-INF/lib/내부에
mybatis-3.5.6.jar 파일을 추가해주었다.

2교시

해당 프로젝트 DB폴더에

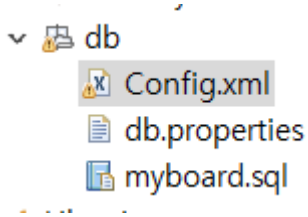
1. db.properties를 생성해준 후

key와 value값을 가지고있는

파일을 생성해 주었다.

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

1. config.xml 이라는 이름을 가진 XML파일을 생성해 주었다.



- 해당 파일을 config파일로 만들어주기 위해서
- https://mybatis.org/mybatis-3/ko/getting-started.html

XML설정파일에서 지정하는 마이바티스의 핵심이 되는 설정은 트랜잭션을 제어하기 위한 TransactionManager과 함께 데이터베이스 Connection인스턴스를 가져오기 위한 DataSource 를 포함한다. 세부적인 설정은 조금 뒤에 보고 간단한 예제를 먼저보자.

```
<?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>
  <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="org/mybatis/example/BlogMapper.xml"/>
```

```
  </mappers>
</configuration>
```

해당 파일의 상단부의 의미

```
<?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">
```

- 지정해줌으로 이 파일은 config파일이다를 명시해준다!

- 다음으로 <configuration></configuration>태그 내부에

```
- <properties resource="db/db.properties"/>를 통하여

<!— db 접속 파일 경로(db.properties) —>
```

- <!-- typeAlias -->

typealias 기존에 선언되어있는 유형에 새로운 유형의 별칭을 사용함으로써 코드를 더 읽기 쉽도록, 이해하기 쉽도록 명확하게 만드는문법입니다.

```
<!-- typeAlias -->
  <typeAliases>
    <typeAlias type="com.my.dto.MyBoardDto" alias="MyBoardDto"/>
  </typeAliases>
```

property 태그 내부에는 properties 파일 의 정보를 가져온다.

db.properties | Config.xml

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

```
<!-- db 접속 설정 -->
  <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>
```

Mapper 경로를 지정해준다

```
<!-- Mapper 경로 -->
  <!-- mappers이기 때문에 mapper파일을 미리 올려두고 경로를 지정해서 잡아준다. -->
  <mappers>
    <mapper resource="db/Mapper.xml"/>
    <mapper resource="db/Mapper2.xml"/>
  </mappers>
```

ex)
게시판 jdbc를 사용하겠다고 하면
mapper.xml을 사용

날씨 jdbc를 사용하겠다고 하면
mapper2.xml를 사용

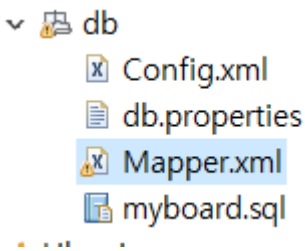
🤠 Mybatis 설정파일 xml에서 validation 에러 발생 처리법의 사본

settings

```
<settings>
    <setting name="jdbcTypeForNull" value="NULL"/>
  </settings>
```

- 해당 셋팅은 만약 null값이 넘어온다면 NULL로 표현해주어라 라는 setting 이다.
- https://mybatis.org/mybatis-3/ko/configuration.html#settings
  - Setting 관련 Mybatis Home Page 이다.

Mapper



다음은 Config.xml에서 Mapper지정을 해준 경로에 Mapper.xml파일을 만들어 주었다. 해당 XML파일에서는 sql문을 저장하고 사용할 것이다!

- sql문을 mapping하겠다

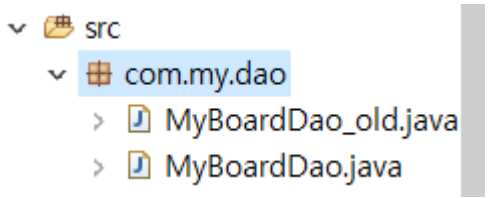
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 해당 파일은 mapper파일이다를 명시해주자! -->

<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
```

```
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.my.myboard">

  <select id="selectAll" resultType="MyBoardDto">
    SELECT MYNO,MYNAME,MYTITLE,MYCONTENT,MYDATE
    FROM MYBOARD
    ORDER BY MYNO DESC
  </select>



</mapper>
```

- 이 때 쿼리문에 ;(세미콜론)을 넣지 않아야 한다. (세미콜론은 쿼리문의 끝을 나타내는 요소이기 때문이다)
- resultType은 어디에 담을것이냐 라는 것을 명시해준다.

  -현재 클래스명만 적은 이유는 TypeAlias에서 별칭을 통하여 패키지명을 별칭으로 적어주었기 때문이다.

DAO를 수정



- 이전 코드를 MyBoardDao_old로 만들고
- 메소드 내부내용을 null로 만들어주었다.

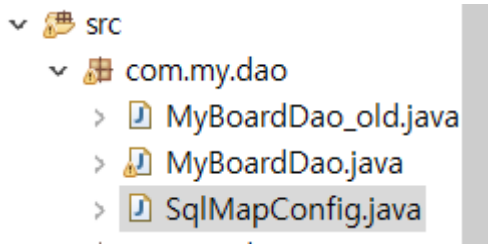
2교시 까지는 설정파일을 만들어 주었다.

3교시 부터는 설정파일을 통해서 쿼리를 실행할 객체를 만들어준다.

3교시

```java
public class MyBoardDao {

    //전체출력
    public List<MyBoardDto> selectAll(){
        List<MyBoardDto> list = new ArrayList<MyBoardDto>(

        SqlSession session = null;
        //import org.apache.ibatis.session.SqlSession;

```

- SqlSession 객체를 생성해준다



- query문을 실행할 수 있도록 SessionFactory를 만들어준다.

- 해당 파일에는 sqlSessionFactory타입의 필드를 지정해주고

```
private SqlSessionFactory sqlSessionFactory;
```

- sqlSessionFactory를 리턴해주는 메소드를 만들어준다.

```
public SqlSessionFactory getsqlSessionFactory()
```

- Config.xml파일의 경로를 잡아주는 변수 resource생성

```
String resource = "db/Config.xml";
```

- Reader를 사용하여 getResourceAsReader(경로);를 작성하고 Config.xml파일의 정보를 읽어온다.

```
Reader reader = Resources.getResourceAsReader(resource);
```

- 읽어온 정보를 가지고 sqlSessionFactory 객체 생성

```
sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);
```

- 전체는 이러하다

```
package com.multi.dao;

import java.io.IOException;
import java.io.Reader;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class SqlMapConfig {
  private SqlSessionFactory sqlSessionFactory;

  public SqlSessionFactory getsqlSessionFactory() {

    try {
      String resource = "db/Config.xml"
      Reader reader = Resources.getResourceAsReader("resource");

      sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);

      reader.close();
    } catch (IOException e) {

      e.printStackTrace();
    }
```

```
    return sqlSessionFactory;
  }

}
```

Dao로 이동하여

```
public class MyBoardDao extends SqlMapConfig {

  //전체출력
  public List<MyBoardDto> selectAll(){
    List<MyBoardDto> res = new ArrayList<MyBoardDto>();

    SqlSession session = null;
    //import org.apache.ibatis.session.SqlSession;

    //session 만들기!
    session = getSqlSessionFactory().openSession(true);

    //해당하는 namespace를 가진 mapper파일의 selectAll이라는 id를 가진것을 실행하겠다.
    //List에 담겨서 리턴을 해주었다!
    res = session.selectList("com.my.myboard.selectAll");

    session.close();

    return res;
  }
```

Dao에서는 해당 두 줄로 변한다!!!!!!!!!

- session 만들어주기

session = getSqlSessionFactory().openSession(true);

이때 true는 Auto commit을 의미한다.

- res = session.selectList("com.my.myboard.selectAll");
  - //해당하는 namespace를 가진 mapper파일의 selectAll이라는 id를 가진것을 실행하겠다.
    //List에 담겨서 리턴을 해주었다!

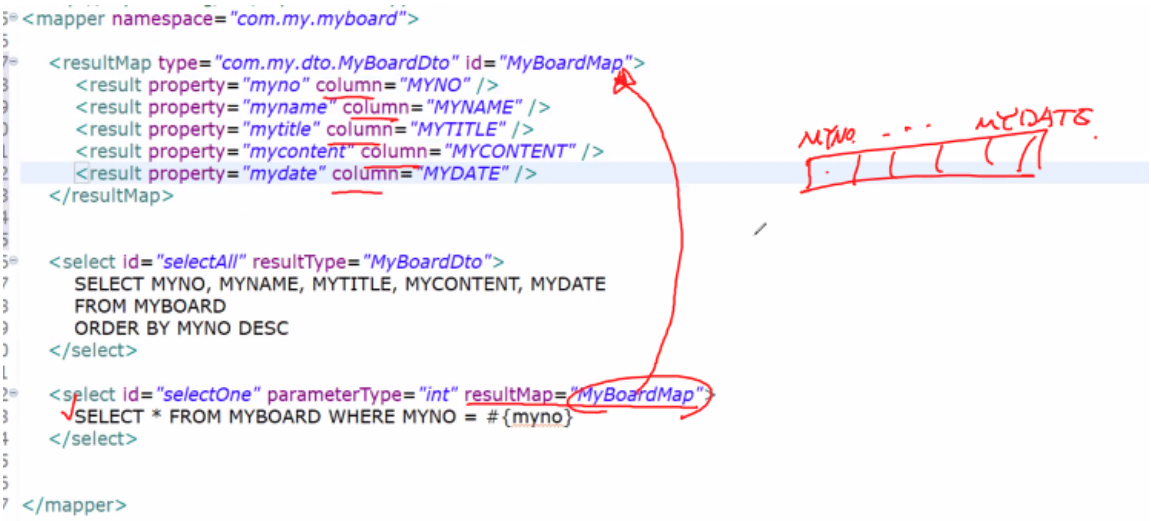
다음은 selectOne 이다

- 설정파일은 모두 준비가되었다! 이제 mapper에서 원하는 쿼리문을 작성하자

```
<select id="selectOne" parameterType="int" resultMap="MyBoardMap">
    SELECT * FROM MYBOARD WHERE MYNO = #{myno}
  </select>
```

- Mybatis에서는 ?대신에 #{000}으로 사용한다.
- selectOne 의 경우에는 parameter 값이 필요하기때문에 parameterType을 명시해주었다.

- 이번에는 resultMap을 통해서 사용해보자




```
<mapper namespace="com.my.myboard">

    <resultMap type="com.my.dto.MyBoardDto" id="MyBoardMap">
        <result property="myno" column="MYNO" />
        <result property="myname" column="MYNAME" />
        <result property="mytitle" column="MYTITLE" />
        <result property="mycontent" column="MYCONTENT" />
        <result property="mydate" column="MYDATE" />
    </resultMap>


    <select id="selectAll" resultType="MyBoardDto">
        SELECT MYNO, MYNAME, MYTITLE, MYCONTENT, MYDATE
        FROM MYBOARD
        ORDER BY MYNO DESC
    </select>

    <select id="selectOne" parameterType="int" resultMap="MyBoardMap">
        SELECT * FROM MYBOARD WHERE MYNO = #{myno}
    </select>


</mapper>
```

- resultMap을통해서 column에다가 담겠다!
- 설정되어있는 해당 컬럼값을 필드에 되돌려주며 사용이가능 하다.
  이부분은 필드명과 컬럼명이 다를때 자동으로 담기지 않게 되는데 이때 resultMap을 통해서
  넣어줄 수 있다.

4교시

- Mapper.xml 을 통해서 지정을 잘 해 주었다면 Dao에서
  - session 객체를 만들어주고 경로를 잡은후에 실행하여준다

```
//선택출력
public MyBoardDto selectOne(int myno) {
    SqlSession session = null;
    MyBoardDto res = null;

    session = getSqlSessionFactory().openSession(true);

    res = session.selectOne(namespace+"selectOne",myno);

    return res;
}
```

- 이때 namespace와 ID를 작성후에 selectOne 이기 때문에 myno를 파라미터 값으로 넘겨서 처리를 해준다

```
    private String namespace ="com.my.myboard.";
```

- namespace를 실행할때마다 작성하기 힘들다면 field를 통해서 작성해주어도 된다.

- 다음은 *Mapper.xml을 통해서*

```
    <insert id="myinsert" parameterType="MyBoardDto">
        INSERT INTO MYBOARD VALUES(MYSEQ.NEXTVAL, #{myname}, #{mytitle},#{mycontent},SYSDAT
    </insert>
</mapper>
```

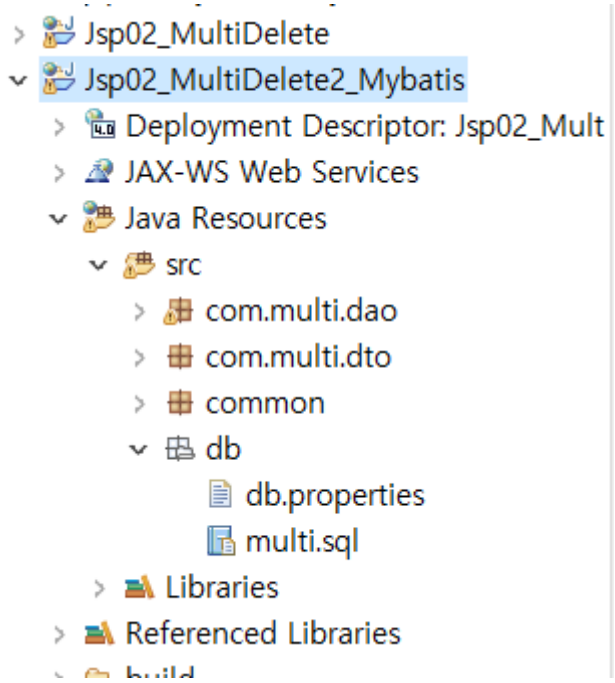
- ID가 myinsert이며 파라미터 타입이 com.my.myboard.MyBoardDto 인 매퍼를 완성시켰다.
- 이때 입력받는 값들은 #{}을 통하여 필드명을 넣어주었다.

```
//추가
public int insert (MyBoardDto dto) {
    SqlSession session = null;
    int res = 0;

    try{
        session = getSqlSessionFactory().openSession(true);
        res = session.insert(namespace+"myinsert",dto);
    }catch(Exception e) {
        e.printStackTrace();

    }finally {
        session.close();
    }



    return res;
}
```

- Dao에서는 session객체를 만들어주고 dto를 파라미터값으로 넘기며 실행한후
  - 결과값을 return 해주고있다.

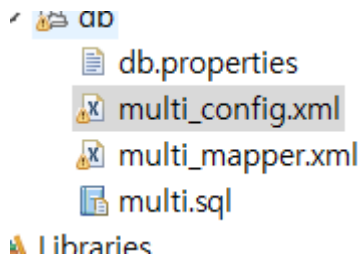
새로운 프로젝트 복사

- Jsp02_MultiDelete2_Mybatis 프로젝트를 생성해준다.

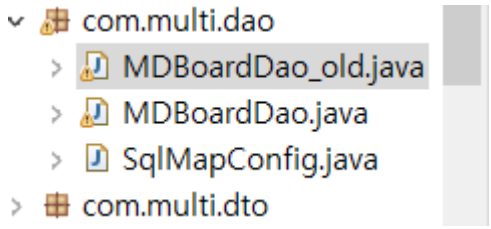


- 드라이버 설정을 위하여 db.properties파일을 생성하고 드라이버 정보를 생성하였다.

- 다음으로 db.multi_config.xml와 db.multi_mapper.xml 파일을 생성해 주었다.



- 필요한 클래스 2개를 생성해 주었다
- com.multi.dao.MDBoardDao_old , com.multidao.SqlMapConfig



- 그 후 동일하게 config.xml과 mapper.xml에

```
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">



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

-각각 작성 해주었다.

숙제

- Mybatis를 사용하여 jsp01 만들기
- MultiBoard_MyBatis 프로젝트 selectOne 만들기