

Real Alternative DBMS ALTIBASE, Since 1999

MyBatis 연동 가이드

2014. 10



Copyright © 2000~2014 ALTIBASE Corporation. All Rights Reserved.

Document Control

Change Record

Date	Author	Change Reference
2013-12	sypark	Created
2014-10	Dbpark	Updated

Reviews

Date	Name (Position)

Distribution

Name	Location

목차

개요	5
MYBATIS 개요	6
MyBATIS 란?	6
MyBatis 다운로드	7
MYBATIS와 IBATIS와의 차이점	8
Java 요구 버전	8
Package 내부 구조의 변경	8
SqlMap.xml 내부 구조의 변경	8
사용 용어의 변경	8
네임스페이스 방식의 변경	8
MYBATIS를 이용한 SAMPLE 작성	9
Mapper 파일 작성	9
Configuration 파일 작성	10
Application 작성	11
ALTIBASE 연동	13
ALTIBASE JDBC Driver 얻는 방법	13
JDBC Driver 에 설정하는 방법	14
Configuration 파일에 dataSource를 설정하여 ALTIBASE와 연동	15
FailOver를 이용한 Connection	16
ALTIBASE5 와 이전 버전을 동시에 Connection	17
Procedure 호출	20
Function 호출	21
MYBATIS, SPRING, ALTIBASE 연동	23
Spring 에 MyBatis를 연동하여 dataSource를 설정	23
Connection Pool 설정 방법	25
트랜잭션 관리	26
MyBatis 에서 트랜잭션 관리	26
MYBATIS 연동 시 고려사항	28
LOB 데이터 처리	28
Insert시 insert query가 중복되어 보내지는 현상	30
부록(MYBATIS-ALTIBASE 연동)	31
DB 테이블 및 시퀀스 생성	31
프로젝트 생성	31
패키지 명명 규칙	32
Configuration 파일 작성	32
Mapper 파일 작성	33
Application 작성	35
관련 JAR 파일 추가	41
Application 실행	42
부록 2(SPRING-MYBATIS-ALTIBASE 연동)	43
DB 테이블 및 시퀀스 생성	43
Spring 설치	43
Maven 설치	44
프로젝트 생성	46
패키지 명명 규칙	47
ApplicationContext 파일 작성	47

<i>Mapper</i> 파일 작성.....	49
Dependency Injection.....	51
Application 작성.....	51
관련 JAR 파일 추가.....	57
Application 실행.....	60

개요

본 문서는 myBatis 환경, ibatis와의 차이점, myBatis에서 ALTIBASE와 연동하는 방법에 대해 기술한다. myBatis 3.2.8, ALTIBASE는 6.3.1 버전, 개발 IDE로는 Eclipse, Maven을 사용하였으며 chapter 와는 별도로 예제가 제공된다.

본 문서와 더불어 개발 시 참고해야 할 문서들은 다음과 같다.

1. 『ALTIBASE 개발가이드』
2. 『JAVA 개발가이드』
3. 『ALTIBASE_JBOSS 연동가이드』
4. 『ALTIBASE_TOMCAT 연동가이드』
5. 『ALTIBASE_WEBSPPHERE 연동가이드』
6. 『ALTIBASE_WEBLOGIC 연동가이드』
7. 『ALTIBASE_Spring 연동가이드』
8. 『ALTIBASE_HIBERNATE 연동가이드』
9. 『ALTIBASE_iBatis 연동가이드』

MyBatis 개요

본 장에서는 MyBatis의 개념과 특징, 다운로드 및 사용 방법에 대해 살펴본다.

MyBatis란?

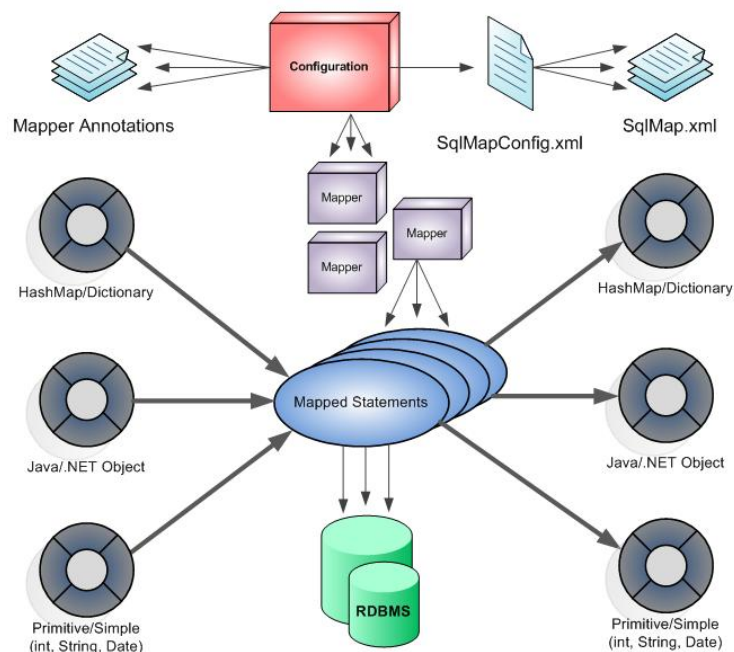
MyBatis 는 개발자가 지정한 SQL, 저장프로시저 그리고 몇가지 고급 매핑을 지원하는 퍼시스턴스 프레임워크이다.

MyBatis 는 JDBC 코드와 수동으로 셋팅하는 파라미터와 결과 매핑을 제거한다.

MyBatis 는 데이터베이스 레코드에 원시타입과 Map 인터페이스 그리고 자바 POJO 를 설정하고 매핑하기 위해 XML 과 애노테이션을 사용할 수 있다.

기존의 JDBC를 이용하여 프로그래밍하는 방식은 프로그램 소스 안에 SQL문을 작성하였지만, MyBatis를 이용하면 SQL문을 프로그램에서 분리하여 XML 파일에 별도로 작성한다. 따라서 프로그래머가 기존의 JDBC를 사용할 때 보다 프로그래밍하는 부담이 줄어들게 된다. 뿐만 아니라 SQL을 변경하고자 할 경우 기존처럼 프로그램을 수정하는 것이 아니라 XML 파일의 SQL문 만을 변경하면 되기 때문에 SQL 변환이 자유롭다는 특징이 있다.

다음은 MyBatis의 구조를 간단하게 표현한 그림이다.



Configuration 파일(SqlMapConfig.xml) : DB 설정과 트랜잭션 등 Mybatis가 동작하는 규칙을 정의.

매퍼(Mapper) : SQL을 XML에 정의한 매퍼 XML 파일과 SQL을 인터페이스의 메소드마다 애노테이션으로 정의한 매퍼 인터페이스를 의미.

매핑 구문(Mapped Statements): 조회 결과를 자바 객체에 설정하는 규칙을 나타내는 결과 매핑과 SQL을 XML에 정의한 매핑 구문을 의미.

매핑 구문을 정의하는 방법은 애노테이션과 XML 방식 두 가지가 존재함.

사용자는 CRUD에 대한 각각의 SQL문은 SqlMap XML 파일에 작성하고 이 파일들을 SqlMapConfig XML 파일에 작성하면 MyBatis API를 통해 자동으로 Mapping된 Statement 객체들을 생성하여 이를 통해 DB에 SQL문을 실행하게 된다.

MyBatis는 현재 구글이 인수하였으며 홈페이지 주소는 다음과 같다.

<http://blog.mybatis.org/>

<http://mybatis.github.io/mybatis-3/> -> MyBatis에 대한 다양한 레퍼런스를 제공하며 한글 번역 페이지도 존재하고 있다. (<http://mybatis.github.io/mybatis-3/ko/>)

MyBatis 다운로드

MyBatis를 사용하기 위해서 Mybatis 관련 jar 파일이 필요하다. 이 jar 파일은 <http://repo1.maven.org/maven2/org/mybatis/mybatis/3.2.8/> 페이지에서 다운로드 받을 수 있다. mybatis-3.2.8.jar를 받아 eclipse에서 추가 해주면 되며 Maven을 사용한다면 다음의 구문을 pom.xml에 추가 해주면 된다.

```
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.2.8</version>
</dependency>
```

Maven을 통하여 Library를 추가하는 방법은 부록 2에서 상세히 설명 한다.

MyBatis와 iBatis와의 차이점

iBatis가 MyBatis로 변경되면서 버전만 변경된 것으로 생각했었으나(iBatis : ~ 2.3 MyBatis : 2.5 ~) 확인해보니 변경점이 많아져서 따로 페이지에 기술하게 되었다.

Java 요구 버전

iBatis에서는 JDK 1.4 이상에서 사용이 가능 하였으나,

MyBatis에서는 JDK 1.5 이상을 요구 한다.(MyBatis 3.2 이상 버전은 JDK 1.6 이상을 요구한다.)

Package 내부 구조의 변경

iBatis : com.ibatis.*

MyBatis : org.apache.ibatis.*(이름은 변경되었지만 내부적으로는 여전히 iBatis를 사용하고 있다.)

SqlMap.xml 내부 구조의 변경

가장 큰 변경점은 parameterMap이 Deprecated된 점이라고 할 수 있다. parameterMap을 사용하지 못하게 되며 기존 parameterMap을 사용하던 부분은 아래와 같이 parameterType에 정의하게 되었다.

```
ex) <insert id="insertBlobData" parameterType="LobVo">
    insert into test_blob(user_no, user_name, blob_data, clob_data, reg_date)
    values(#{userNo}, #{userName}, #{blobData,jdbcType=BLOB},
    #{clobData,jdbcType=CLOB}, #{regDate})
</insert>
```

사용 용어의 변경

SqlMapConfig -> Configuration

sqlMap -> mapper로 변경됨.

네임스페이스 방식의 변경

sqlMap 별로 줄여놓은 이름을 사용할 수 없게 되며 경로를 모두 명시해 주어야 함.

또한 iBatis에서는 namespace가 선택이었지만 MyBatis는 필수 항목이다.

Ex) iBatis : <sqlMap namespace="Lob">

MyBatis : <mapper namespace="com.altibase.sample.mapper.LobMapper">

MyBatis를 이용한 sample 작성

MyBatis를 이용하여 SQL문을 처리하기 위해서는 Configuration XML 파일과 Mapper XML 파일을 작성해야 한다. 이 파일들은 프로그래머에게 JavaBean을 PreparedStatement의 파라미터와 ResultSet으로 쉽게 mapping 할 수 있도록 해준다. 본 장에서는 Configuration XML 파일, Mapper XML 파일을 작성하는 방법과 application에서 이 파일을 이용하여 실제로 SQL을 처리하는 방법에 대해 설명한다. Sample 프로그램을 작성하는 보다 자세한 내용은 부록 부분을 참고하면 된다.

Mapper 파일 작성

Mapper XML 파일은 DB로 전송할 SQL 구문, PreparedStatement로 binding될 parameter의 mapping, ResultSet의 result의 mapping들을 명시하는 파일이다.

다음은 person 테이블에 CRUD를 처리하는 Mapper XML 파일을 작성한 예제이다. (UserMapper.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.altibase.sidu.mapper.UserMapper">
  <select id="selectUserData" parameterType="Integer" resultType="User">
    SELECT user_no as userNo,
    user_name as userName,
    user_content as userContent,
    reg_date as regDate
    FROM users
    WHERE user_no = #{userNo}
  </select>

  <select id="selectAllUserData" resultType="User">
    SELECT user_no as userNo,
    user_name as userName,
    user_content as userContent,
    reg_date as regDate
    FROM users
  </select>

  <insert id="insertUserData" parameterType="User">
    insert into
    users(user_no, user_name, user_content, reg_date)
    values(#{userNo}, #{userName}, #{userContent}, #{regDate})
  </insert>

  <update id="updateUserData" parameterType="User">
    update users
    set user_name = #{userName},
    user_content = #{userContent},
    reg_date = #{regDate}
    where user_no = #{userNo}
  </update>

  <delete id="deleteUserData" parameterType="User">
    delete from users
    where user_no = #{userNo}
  </delete>
</mapper>
```

```
</mapper>
```

<insert>, <update>, <delete>, <select> 태그에는 CRUD의 작업에 대한 각각의 SQL문을 정의한다.

각각의 태그에 대한 보다 자세한 설명은 <http://mybatis.github.io/mybatis-3/ko/> 사이트를 참고하거나 첨부된 문서 MyBatis-3-User-Guide_ko.pdf 파일을 참고하면 된다.

Configuration 파일 작성

Configuration 파일은 DB 연결을 위한 dataSource, Mapper 파일의 경로, typeAliases의 설정 외 SqlMapClient를 제어할 property들을 작성하는 SQL Maps 설정파일이다.

다음은 Configuration 파일(mybatis-config.xml) 예제이다.

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

<typeAliases>
  <typeAlias type="com.altibase.sidu.model.UserVo" alias="User" />
</typeAliases>

<!-- DB 연결 옵션 : UNPOOLED/POOLED/JNDI -->
<!-- transactionManager 옵션 : JDBC/MANAGED -->
<environments default="development">
  <environment id="development">
    <transactionManager type="JDBC" />
    <dataSource type="POOLED">
      <property name="driver" value="${jdbc.driver}" />
      <property name="url" value="${jdbc.url}" />
      <property name="username" value="${jdbc.username}" />
      <property name="password" value="${jdbc.password}" />
      <property name="poolPingQuery" value="select 1 from dual" />
      <property name="poolMaximumActiveConnections"
value="100" />
      <property name="poolMaximumIdleConnections" value="50" />
      <property name="poolMaximumCheckoutTime" value="20000" />
    </dataSource>
  </environment>
</environments>

<mappers>
  <mapper resource="com/altibase/sidu/mapper/UserMapper.xml" />
</mappers>

</configuration>
```

<properties> 태그에는 name=value 형태로 정의된 property들을 작성한 properties 파일의 경로 및 이름을 명시해주고, <settings> 태그에는 Configuration을 제어할 property들을, <transactionManager> 와 <dataSource>에는 연결할 DB정보를 작성한다. 또, <mappers> 태그에는 미리 작성한 Mapper 파일들의 경로 및 이름을 작성한다.

다만 이전 iBatis와 다른 점은 iBatis에서 DB 설정을 한 개만 할 수 있었던 것에 비해서 MyBatis는 설정 파일에 여러 개의 DB를 설정하고 SqlSessionFactory 객체를 생성하는 시점에서 특정 DB를 설정할 수 있다.

각각의 태그에 대한 보다 자세한 설명은 <http://mybatis.github.io/mybatis-3/ko/> 사이트를 참고하거나 첨부된 문서 MyBatis-3-User-Guide_ko.pdf 파일을 참고하면 된다.

Application 작성

Application에서 MyBatis에서 제공하는 SqlSession 객체의 instance를 이용하여 DB 테이블에 Mapping된 객체와 연동하여 CRUD 작업을 처리할 수 있다.

MyBatis를 이용하여 DB와 연동하기 위해서는 먼저 DB 연결에 대한 설정이 있는 Configuration 파일을 읽어 SqlSessionFactory 객체를 생성하고 해당 객체에서 SqlSession 객체를 얻어 Mapper에 매핑되는 쿼리를 처리하게 된다.

다음은 DB의 users 테이블에 데이터를 삽입, 변경, 삭제, 조회하는 application이다.

Application에서 연결을 얻어오는 부분은 별도의 Class로 분류 하였다.

예) altibase_mybatis_sidu의 SiduMain.java

```
// Configuration DB Connection
MybatisUtil.java
...
InputStream = Resources.getResourceAsStream("mybatis-config.xml");
sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);
sqlSession = sqlSessionFactory.openSession(false);
...
UserVo userVo = new UserVo();
...

//select User
String statement = "com.altibase.sidu.mapper.UserMapper.selectUserData";
userVo = MybatisUtil.getSqlSessionFactory().selectOne(statement, user_no);
...

// select all User
List<UserVo> userVos = MybatisUtil.getSqlSessionFactory().selectList(statement);
...

// update User
String statement = "com.altibase.sidu.mapper.UserMapper.updateUserData";
res_count =
MybatisUtil.getSqlSessionFactory().update(statement, userVo);
...

// insert User
String statement = "com.altibase.sidu.mapper.UserMapper.insertUserData";
res_count = MybatisUtil.getSqlSessionFactory().insert(statement, userVo);
...

// delete User
String statement = "com.altibase.sidu.mapper.UserMapper.deleteUserData";
res_count = MybatisUtil.getSqlSessionFactory().insert(statement, user_no);
```

먼저 Mapper 파일을 읽어 들여 SqlSessionFactory 객체를 생성하며 생성된 SqlSessionFactory 객체에서 SqlSession 객체를 생성 하며,

Sample에서 SqlSession을 가져온 메소드는 다음과 같다.

```
sqlSessionFactory.openSession(boolean arg0);
```

해당 메소드의 Boolean 인자는 autocommit 모드를 결정하는 값이다.(

```
inputStream = Resources.getResourceAsStream("mybatis-config.xml");
```

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

```
sqlSession = sqlSessionFactory.openSession(false);
```

)

이후 SqlSession 객체와 Mapper에 명명되어 있는 id를 가지고 CRUD 클래스의 각각의 메소드를 호출한다. (

```
String statement = "com.altibase.sidu.mapper.UserMapper.updateUserData";
```

```
sqlSession.update(statement, update);)
```

각각의 태그에 대한 보다 자세한 설명은 <http://mybatis.github.io/mybatis-3/ko/> 사이트를 참고하거나 첨부된 문서 MyBatis-3-User-Guide_ko.pdf 파일을 참고하면 된다.

ALTIBASE 연동

MyBatis에서 ALTIBASE를 연동하는 위해서는 ALTIBASE JDBC Driver를 setting하고 Configuration 파일에 ALTIBASE 를 위한 dataSource를 지정하면 된다. 본 장에서는 ALTIBASE JDBC Driver를 얻는 방법, JDBC Driver를 설정하는 방법, Configuration에 dataSource를 설정하는 방법에 대해 설명한다. 또한, FailOver 기능을 사용하는 방법, 여러 버전의 ALTIBASE와 연동하는 방법, Stored Procedure/Function을 호출하는 방법에 대해서도 살펴본다.

ALTIBASE JDBC Driver 얻는 방법

ALTIBASE에서 제공하는 JDBC driver는 Altibase.jar 파일이다. 이 파일은 ALTIBASE가 설치되어있는 서버의 \$ALTIBASE_HOME/lib 디렉토리 안에 존재한다.

ALTIBASE 5 버전부터는 \$ALTIBASE_HOME/lib 디렉토리에 Altibase.jar 파일과 Altibase5.jar 파일이 존재하는데, Altibase.jar는 일반 JDBC Driver 파일이고, Altibase5.jar는 ALTIBASE 5 버전과 그 이하의 버전을 함께 연동하고 싶을 때 사용하는 JDBC Driver 파일이다. 따라서 하나의 ALTIBASE DB와 연동하거나, 또는 버전이 동일한 여러 대의 ALTIBASE와 연동할 경우에는 \$ALTIBASE_HOME/lib/Altibase.jar 파일을 사용하면 된다.

연동하려는 ALTIBASE DB Server와 ALTIBASE JDBC Driver가 호환 가능한지 확인을 위해 ALTIBASE JDBC Driver 버전 확인이 필요하다.

ALTIBASE JDBC Driver 버전을 확인하는 방법은 다음의 명령어를 수행하면 된다.

```
$ java -jar Altibase.jar
JDBC Driver Info : Altibase Ver = 6.3.1.2.6 for JVM v1.4, CMP:7.1.1, Oct  6 2014 13:54:56
```

이때, ALTIBASE DB Server의 cm protocol version과 ALTIBASE JDBC Driver의 CMP가 동일하면 호환 가능하다.

```
$ altibase -v
version 6.3.1.2.7 X86_64_LINUX_redhat_Enterprise_ES4-64bit-6.3.1.2.7-release-GCC3.4.6
(x86_64-unknown-linux-gnu) Oct  8 2014 09:16:24, binary db version 6.2.1, meta version 6.3.1,
cm protocol version 7.1.1, replication protocol version 7.4.1
```

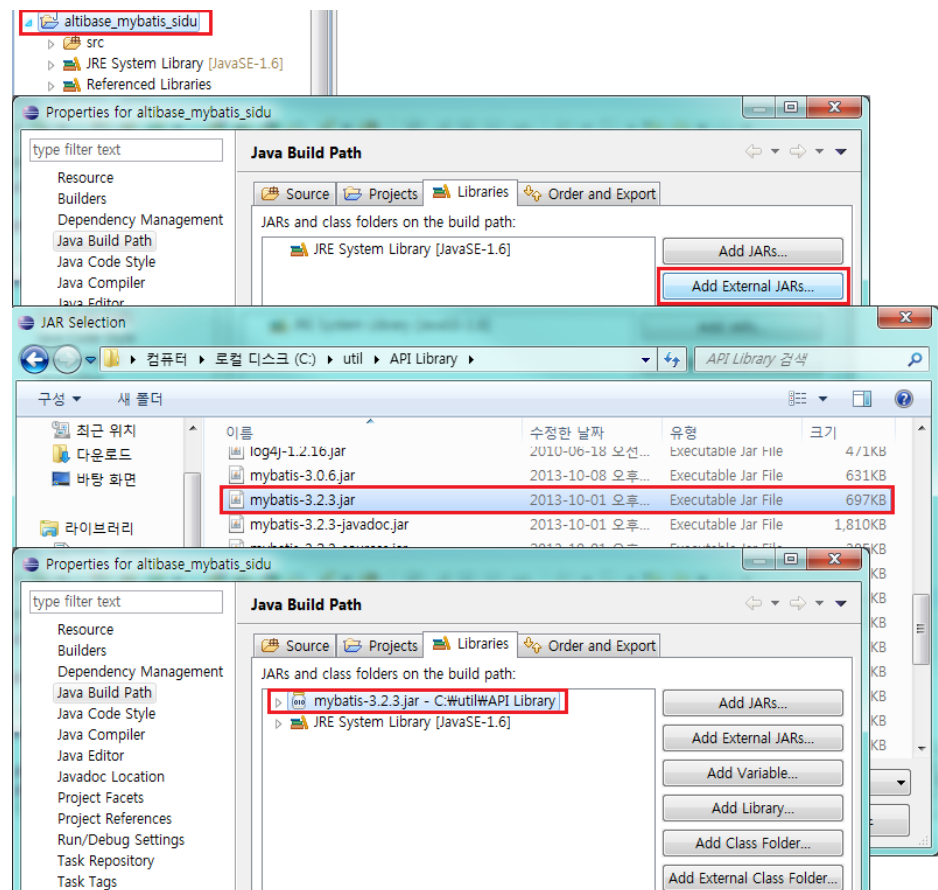
버전이 UP 되면서 JDBC 관련 버그가 fix되었을 가능성이 있으므로, 일반적으로 ALTIBASE DB Server의 버전과 같거나 이 보다 더 최신의 ALTIBASE JDBC Driver 파일을 사용하는 것을 권장한다.

JDBC Driver에 설정하는 방법

다운로드 받은 JDBC Driver 인 Altibase.jar 파일은 classpath에 추가하거나 웹서버의 적절한 디렉토리에 위치시킨다.

만약, Eclipse를 사용하여 개발한다면 다음과 같이 해당 프로젝트에 ALTIBASE JDBC Driver를 추가할 수 있다.

프로젝트 - Properties - Java Build Path 메뉴로 이동하여 Add External JARS 메뉴를 통해 라이브러리를 등록 한다.



Configuration 파일에 dataSource를 설정하여 ALTIBASE와 연동

Configuration 파일의 <configuration> 태그에 ALTIBASE 용 property를 지정하여 ALTIBASE와 연결하면 된다. 이 때 Configuration 파일에 직접 property 값을 입력할 수 있고, 또는 별도의 properties 파일을 작성하여 이 파일에 작성된 property값을 로딩하여 사용할 수 도 있다.

다음은 db.properties 라는 properties 파일에 ALTIBASE에 대한 property들을 정의하고, 이 property들을 읽어와 Configuration 파일에서 사용하는 예제이다.

예) altibase_mybatis_sidu의 db.properties 파일

```
jdbc.driver=Altibase.jdbc.driver.AltibaseDriver
jdbc.url=jdbc:Altibase://192.168.1.35:36492/mydb
jdbc.username=sys
jdbc.password=manager
```

이 파일에 설정된 각각의 값의 의미는 다음과 같다.

Property	설명
driver	ALTIBASE JDBC driver class Name
url	ALTIBASE와 연결을 위한 Connection string정보 jdbc:Altibase://IP:port_no/db_name" 형태로 기입
username	데이터베이스 계정
password	데이터베이스 패스워드

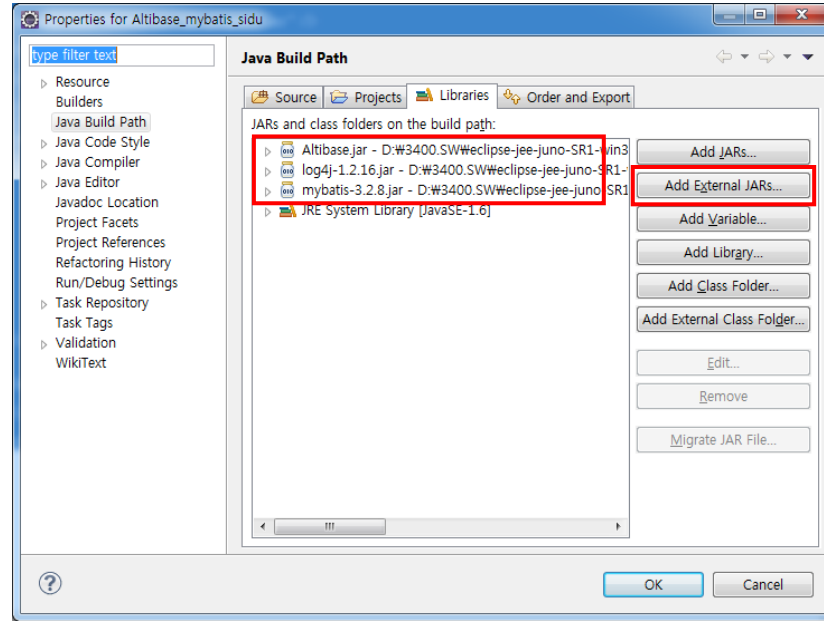
예) altibase_mybatis_sidu의 mybatis-config.xml 파일

```
<configuration>
  <properties resource="db.properties" />
  <typeAliases>
    <typeAlias type="com.altibase.sidu.model.UserVo" alias="User" />
  </typeAliases>

  <!-- DB 연결 옵션 : UNPOOLED/POOLED/JNDI -->
  <!-- transactionManager 옵션 : JDBC/MANAGED -->
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
        <property name="driver" value="{jdbc.driver}" />
        <property name="url" value="{jdbc.url}" />
        <property name="username" value="{jdbc.username}" />
        <property name="password" value="{jdbc.password}" />
        <property name="poolPingQuery" value="select 1 from dual" />
        <property name="poolMaximumActiveConnections" value="100" />
        <property name="poolMaximumIdleConnections" value="50" />
        <property name="poolMaximumCheckoutTime" value="20000" />
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="com/altibase/sidu/mapper/UserMapper.xml" />
  </mappers>
</configuration>
```

db.properties에 지정한 driver, url, username, password property들을 읽어와 dataSource의 JDBC.Driver, JDBC.ConnectionURL, JDBC.Username, JDBC.password property에 setting하고 있다.

위의 예제 SimpleConnection 프로젝트를 실행하기 위해서는 Altibase.jar, mybatis-3.2.8.jar 파일이 필요하다.



FailOver를 이용한 Connection

ALTIBASE 5.3.3 부터 FailOver를 지원하는데, FailOver 기능을 사용하기 위해서는 dataSource의 Connection url을 적어주는 부분에 FailOver 관련 속성을 넣어주면 된다.

다음은 FailOver를 이용하여 ALTIBASE에 연결하는 예제이다. db.properties 파일에 Connection url 부분을 정의하였다.

예) FailOverSample의 db.properties 파일

```
driver=Altibase.jdbc.driver.AltibaseDriver
url=jdbc:Altibase://192.168.6.224:21129/mydb?
  AlternateServers=(192.168.1.35:21129)&
  ConnectionRetryCount=1&ConnectionRetryDelay=1&
  SessionFailOver=on&LoadBalance=off&Healthcheckduration=10&
  Failover_source=MESSAGE&
username=sys
password=manager
```

위의 파일에 지정한 Connection url 부분에 정의할 수 있는 FailOver 관련 property는 다음과 같다.

Property	설명
AlternateServer	장애 발생시 접속하게 될 가용 서버를 나타내며 (IP Address1:Port1, IP Address2:Port2,...) 형식으로 기술한다.
ConnectionRetryCount	가용 서버 접속 실패 시, 접속 시도 반복 횟수

Property	설명
ConnectionRetryDelay	가용 서버 접속 실패 시, 다시 접속을 시도하기 전에 대기하는 시간(초 단위)
LoadBalance	on으로 설정하면 최초 접속 시도 시에 기본 서버와 가용 서버를 포함하여 랜덤으로 선택한다. off로 설정하면 최초 접속 시도 시에 기본 서버에 접속하고, 접속에 실패하면 AlternateServer로 기술한 서버에 접속한다.
SessionFailOver	STF(Service Time Fail-Over)를 할 것인지 여부를 나타낸다. on : STF, off : CTF CTF(Connection Time Fail-Over)는 DBMS 접속 시점에 장애를 인식하여 다른 정상서버로 접속을 재시도하는 것을 의미한다. STF(Service Time Fail-Over)는 서비스하는 도중에 장애를 감지하여 다른 가용 노드의 DBMS에 다시 접속하여 세션의 프로퍼티를 복구한 후 사용자 응용 프로그램의 업무 로직을 다시 수행할 수 있도록 하는 것을 의미한다. (STF는 DB접속에 대해서만 Fail-Over를 수행해주는 것이며 실패한 트랜잭션에 대해서는 사용자에게 의해 재처리되어야 한다)
Healthcheckduration	Failover가 발생한 서버가 다시 AlternativeServer 목록으로 재설정 되기 위한 대기 시간을 지정한다. 서버에 Connection Failover가 발생하면, 해당 서버는 AlternativeServer의 목록에서 제거된다. 그리고 삭제된 서버는 특정 시간이 지난 후에 다시 AlternativeServer 목록에 추가되는데, 그 시간을 결정하는 프로퍼티가 HealthCheckDuration이다. 단위는 초(second)이다.
Failover_source	Failover를 수행할 때, 서버에 전달하는 Failover source에 대한 설명을 지정한다. 이 정보는 V\$SESSION 성능 뷰의 FAILOVER_SOURCE 칼럼에 저장된다.

위의 예제 FailOverSample 프로젝트를 실행하기 위해서는 "Configuration 파일에 dataSource를 설정하여 ALTIBASE와 연동"과 마찬가지로 Altibase.jar, ibatis-2.3.4.x.jar 파일이 필요하다.

ALTIBASE5 와 이전 버전을 동시에 Connection

ALTIBASE 5 부터는 하나의 어플리케이션에서 ALTIBASE 6 와 ALTIBASE 5 혹은 ALTIBASE 4 와 동시에 연결할 수 있도록 ALTIBASE 5 버전 전용의 JDBC Driver(Altibase5.jar)를 제공한다. 이 Driver를 이용하면 ALTIBASE 6 - ALTIBASE 5, 혹은 ALTIBASE 6 - ALTIBASE 4, ALTIBASE 5.1.5 - ALTIBASE 5.3.3 간 두 버전의 ALTIBASE에 접속이 가능하다.

기존의 Altibase.jar와 구별하기 위해 별도로 ALTIBASE 5 전용의 Altibase5.jar 가 필요하다. 또한 dataSource에 지정해주는 부분에 JDBC Driver 클래스 이름도 기존의

Altibase.jdbc.driver.AltibaseDriver 대신 ALTIBASE 5 전용의 Altibase5.jdbc.driver.AltibaseDriver를 지정해야 한다.

MyBatis에 다른 버전의 ALTIBASE 와 연동하기 위해서는 각 버전에 해당하는 <environment> 태그를 작성하여 Application에서 <environment>의 id를 통해 DB 연결 시 각각의 설정을 읽어드리면 된다. 이 때 주의할 점은 프로그램에서 Altibase5.jdbc.driver.AltibaseDriver를 먼저 로딩한 후에 Altibase.jdbc.driver.AltibaseDriver를 로딩해야 한다는 것이다.

다음은 Altibase.jar와 Altibase5.jar 파일을 이용하여 두 버전의 ALTIBASE의 드라이버를 로딩하는 예제이다.

예) altibase_mybatis_MultiVersionConnection의 db.properties 파일

ALTIBASE 6 버전에 대한 설정

```
# Altibase □
jdbc.driver=Altibase.jdbc.driver.AltibaseDriver
jdbc.url=jdbc:Altibase://192.168.1.62:21020/mydb
jdbc.username=test
jdbc.password=test
```

예) altibase_mybatis_MultiVersionConnection의 db.properties 파일

ALTIBASE 6 이전 버전에 대한 설정

```
# Altibase 5□
jdbc5.driver=Altibase5.jdbc.driver.AltibaseDriver
jdbc5.url=jdbc:Altibase://192.168.1.147:21020/mydb
jdbc5.username=test
jdbc5.password=test
```

예) altibase_mybatis_MultiVersionConnection의 mybatis-config.xml파일

ALTIBASE 6 버전에 대한 설정

```
<environment id="release">
  <transactionManager type="JDBC" />
  <dataSource type="POOLED">
    <property name="driver" value="${jdbc.driver}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
    <property name="poolPingQuery" value="select 1 from dual" />
    <property name="poolMaximumActiveConnections" value="100" />
    <property name="poolMaximumIdleConnections" value="50" />
    <property name="poolMaximumCheckoutTime" value="20000" />
  </dataSource>
</environment>
```

ALTIBASE 6 이전 버전에 대한 설정

```
<environment id="development">
  <transactionManager type="JDBC" />
  <dataSource type="POOLED">
    <property name="driver" value="${jdbc5.driver}" />
    <property name="url" value="${jdbc5.url}" />
    <property name="username" value="${jdbc5.username}" />
    <property name="password" value="${jdbc5.password}" />
  </dataSource>
</environment>
```

```

<property name="poolPingQuery" value="select 1 from dual" />
<property name="poolMaximumActiveConnections" value="100" />
<property name="poolMaximumIdleConnections" value="50" />
<property name="poolMaximumCheckoutTime" value="20000" />
</dataSource>
</environment>

```

예) altibase_mybatis_MultiVersionConnection의 MultiVersionConnectionMain.java파일

```

...
MultiVersionConnectionSelect multi_connection_select =
new MultiVersionConnectionSelect();

System.out.println("DB Version = " +
multi_connection_select.MultiVersionConnectionDBVersionSelect("release"));

System.out.println("DB Version = " +
multi_connection_select.MultiVersionConnectionDBVersionSelect("development"));
...

```

altibase_mybatis_MultiVersionConnection의 MultiVersionConnectionSelect.java
(public String MultiVersionConnectionDBVersionSelect(String conn_type))

```

...
String statement =
"com.altibase.multiVersionConnection.mapper.UserMapper.selectDBVersion";
res_string = MybatisUtil.getSqlSessionFactory(conn_type).selectOne(statement);
...

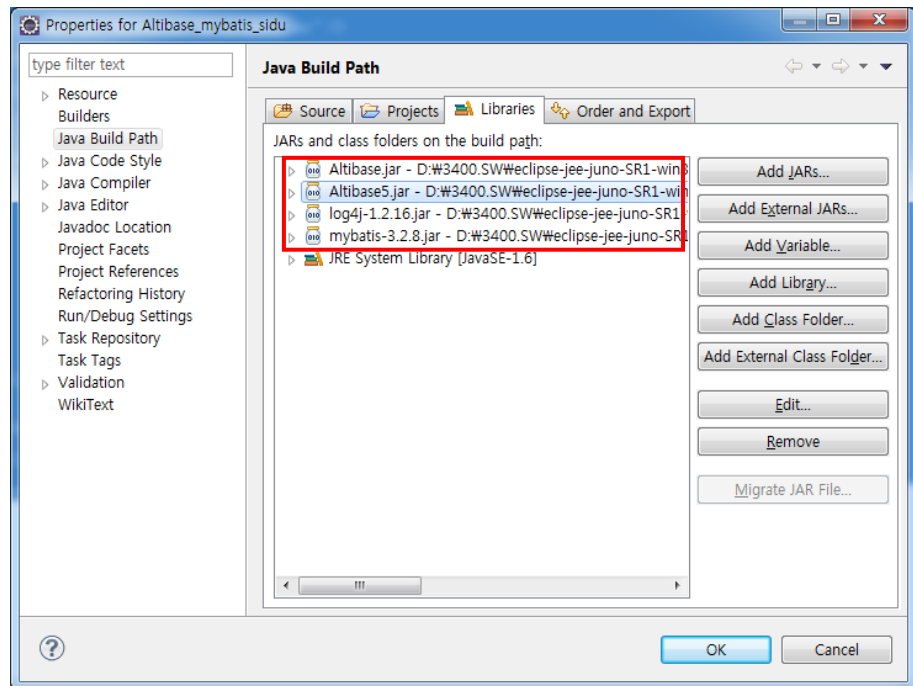
```

위의 MultiVersionConnectionMain.java.java 예제를 보면

Altibase5.jdbc.driver.AltibaseDriver를 Altibase.jdbc.driver.AltibaseDriver보다 먼저 로딩하기 위해 ALTIBASE 5 버전 driver의 <environment> 태그의 id인 release를 인자 값으로 설정하고 있다. 위에서도 기술하였듯이 Altibase5.jdbc.driver.AltibaseDriver 드라이버를 먼저 로딩해야 한다.

예제에 포함된 MultiVersionConnection 프로젝트를 실행하기 위해서는 기존에 사용했던 mybatis.3.2.8.jar 파일 뿐만 아니라, Altibase.jar와 Altibase5.jar 파일이 더 필요하다. 이 파일들은 ALTIBASE가 설치된 디렉토리(\$ALTIBASE_HOME)의 lib 디렉토리 안에 존재하는데 ALTIBASE 5 버전의 Altibase5.jar 파일, 그 이전 버전의 Altibase.jar 파일을 사용하면 된다.

- ALTIBASE 6.3.1 jar : Altibase.jar
- ALTIBASE 5.x.x.jar : Altibase5.jar



Procedure 호출

myBatis에서 DB에 생성한 Stored Procedure을 호출할 경우에는 기본적인 DML 동작과 같이 Configuration 파일에 설정해주면 되며 다음의 항목을 아래와 같은 점에 주의하여 사용 하여야 한다.

- 기존 iBatis에서 사용하던 <procedure> 태그가 사라지고 statementType으로만 판단하므로 Procedure/Function 사용시에는 statementType을 필히 CALLABLE로 설정해야 함.
- 기존.ibatis에서 사용하던 <parameterMap>을 사용할 수 없다.

다음은 Stored Procedure의 Select 하는 예제이다.

예) altibase_mybatis_procedure의 Procedure 생성 구문

```
CREATE OR REPLACE PROCEDURE PROC_SEL_TEST(usr_no IN INTEGER,
res_user OUT TEST_TYPE.TEST_CUR)
as
    SQL_STMT VARCHAR(200);
BEGIN
    SQL_STMT := 'SELECT user_no as userNo,
                user_name as userName,
                user_content as userContent,
                reg_date as regDate
FROM users
WHERE user_no = ?';
    OPEN res_user FOR SQL_STMT USING usr_no;
END;
/

CREATE or replace TYPESET TEST_TYPE
AS
TYPE TEST_CUR IS REF CURSOR;
```

```
END;  
/
```

예) altibase_mybatis_procedure의 UserMapper.xml(Mapper) 파일

```
<select id="testSelectProc" statementType="CALLABLE" parameterType="INTEGER"  
resultType="User">  
  { call PROC_SEL_TEST(  
    #{userNo,mode=IN,jdbcType=INTEGER,javaType=INTEGER}  
  )}  
</select>
```

<select> 태그에 statementType의 속성을 CALLABLE로 지정하여 해당 요청이 Procedure/Function이라는 것을 명시하고 호출하는 Procedure/Function의 IN 파라미터에 대한 설정을 정의하며 resultType에 Bean 클래스를 지정함으로써 해당 procedure에 대한 결과를 Bean 클래스로 받아오게 된다.

이 때 Bean 클래스의 컬럼과 DB서 select하는 컬럼명을 동일하게 맞춰줘야 Bean 클래스에 결과가 리턴 된다.

altibase_mybatis_procedure 예제를 실행하기 위해서는 "Configuration 파일에 dataSource를 설정하여 ALTIBASE와 연동"과 마찬가지로 Altibase.jar, mybatis.3.2.8.jar 파일이 필요하다.

Function 호출

myBatis에서 DB에 생성한 Function을 호출할 경우에는 기본적인 DML 동작과 같이 Configuration 파일에 설정해주면 되며 다음의 항목을 아래와 같은 점에 주의하여 사용 하여야 한다.

- 기존 iBatis에서 사용하던 <procedure> 태그가 사라지고 statementType으로만 판단하므로 Procedure/Function 사용시에는 statementType을 필히 CALLABLE로 설정해야 함.
- 기존.ibatis에서 사용하던 <parameterMap>을 사용할 수 없다.

다음은 단순한 덧셈 결과를 리턴하는 Stored Function 처리 예제이며 기본적인 것은 Stored Procedure와 동일하지만 예제 호출하는 부분이 약간 달라 따로 기술하게 되었다.

예) altibase_mybatis_procedure의 Function 생성 구문

```
CREATE OR REPLACE FUNCTION sum_func  
(  
  p_num1 IN NUMBER,  
  p_num2 IN NUMBER  
)  
RETURN NUMBER  
AS  
  v_num NUMBER;  
BEGIN  
  v_num := p_num1 + p_num2;  
  RETURN v_num;  
END;  
/
```

예) altibase_mybatis_procedure의 UserMapper.xml(Mapper) 파일

```
<select id="testSelectFunc" statementType="CALLABLE" parameterType="map"
resultType="INTEGER">
  { call #{resNum,mode=OUT,jdbcType=INTEGER,javaType=INTEGER} :=
SUM_FUNC(
  #{Num1,mode=IN,jdbcType=INTEGER,javaType=INTEGER},
  #{Num2,mode=IN,jdbcType=INTEGER,javaType=INTEGER}
  )}
</select>
```

기본적인 설정법은 Stored Procedure 방법과 동일 하지만 Mapper에서 Function을 호출하는 형식이 다르다.

Stored Procedure의 호출과의 차이점은 다음과 같다.

Procedure : call PROC_SEL_TEST(#{IN});
Function : call #{OUT} := SUM_FUNC(#{IN})

Function을 Mapper에서 호출 시에 위와 같은 형식을 지켜야 정상적인 Function Call이 가능하다.

altibase_mybatis_procedure 예제를 실행하기 위해서는 "Configuration 파일에 dataSource를 설정하여 ALTIBASE와 연동"과 마찬가지로 Altibase.jar, mybatis.3.2.8.jar 파일이 필요하다.

MyBatis, Spring, ALTIBASE 연동

MyBatis, Spring을 ALTIBASE와 연동하기 위해서는 Spring에 MyBatis 모듈을 이용하여 dataSource를 정의하고 ALTIBASE와 연동해야 한다. 본 장은 Spring에 MyBatis 모듈을 이용하여 ALTIBASE와 연결하는 방법을 설명 한다.

Spring에 MyBatis를 연동하여 dataSource를 설정

Spring과 MyBatis를 함께 사용하여 ALTIBASE를 사용하기 위해서는 Spring의 설정 파일(applicationContext.xml)에 다음의 내역을 설정 한다.

- MyBatis 모듈을 이용한 SqlSessionFactory bean
- 트랜잭션에 대한 설정인 transactionManager
- DB에 대한 dataSource
- Mapper Bean 설정

연동 작업을 수행하려면 Mybatis와 Spring 연동 관련 모듈이 필요하며 모듈의 이름은 다음과 같다.

- mybatis-spring-1.x.x.jar

mybatis-spring 연동모듈을 사용하려면 자바 1.5 이상의 버전이 필요하고 MyBatis와 Spring은 각각 Version 별로 조금씩 다르다. 세부정보는 아래의 표를 참고하기 바란다.

MyBatis Spring 연동모듈	Mybatis	Spring
1.0.0 그리고 1.0.1	3.0.1 에서 3.0.5 까지	3.0.0 또는 그 이상
1.0.2	3.0.6	3.0.0 또는 그 이상
1.1.0 또는 그 이상	3.1.0 또는 그 이상	3.0.0 또는 그 이상

MyBatis와 Spring을 함께 사용하기 위해서는 Spring의 applicationContext.xml 파일에 MyBatis의 SqlSessionFactory bean 및 Mapper bean을 지정해 주면 된다. Mapper bean은 실제 쿼리를 처리하는 메소드를 정의한 bean으로 Spring의 처리 방식인 Interface를 통하여 처리 하게 된다.

MyBatis, Spring을 연동한 환경에서 Spring에 dataSource를 설정하는 방법은 『ALTIBASE_Spring_연동가이드』 문서에서 설명한 방법들 중 하나를 선택하여 SqlSessionFactory bean 및 TransactionManager bean의 dataSource property에 설정한 DB의 datasource명을 기술 하면 된다.

다음은 applicationContext.xml 파일에서 dataSource와 SqlSessionFactory bean, TransactionManager bean을 지정해주는 예제이다.

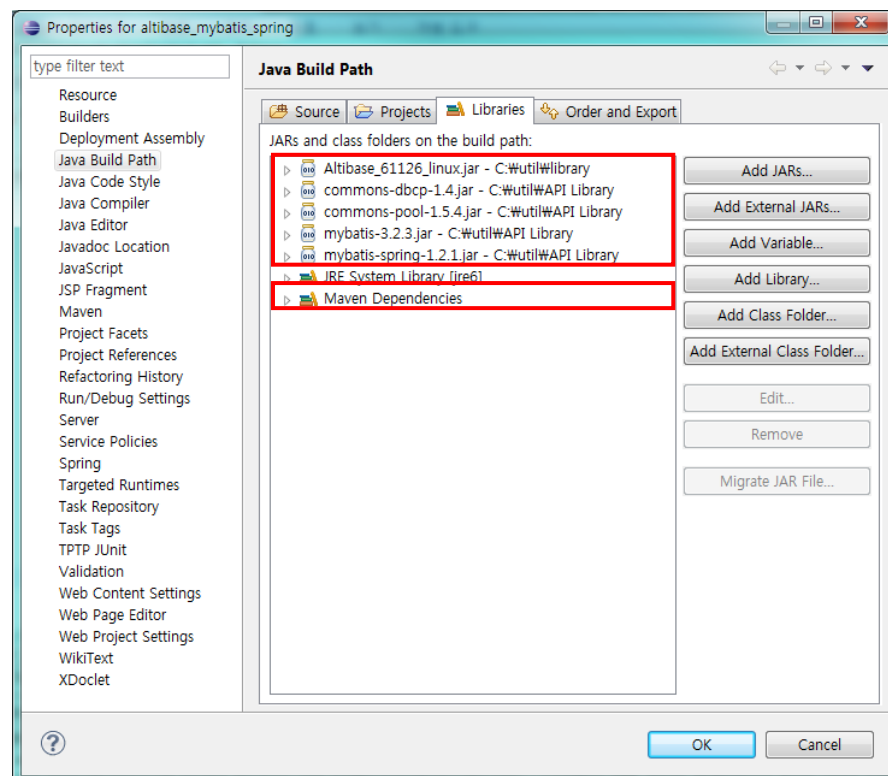
예) altibase_mybatis_spring의 applicationContext.xml 파일

```

...
<!-- DriverManagerDataSource 클래스를 이용한 데이터 소스 설정 -->
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName" value="Altibase.jdbc.driver.AltibaseDriver" />
  <property name="url" value="jdbc:Altibase://192.168.1.35:36492/mydb" />
  <property name="username" value="sys" />
  <property name="password" value="manager" />
</bean>
...
<!-- TransactionManager bean 설정 -->
<bean id="TransactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource" />
</bean>
...
<!-- SqlSessionFactory bean 설정 -->
<bean id="SqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="typeAliasesPackage" value="com.altibase.test.domain" />
</bean>

```

해당 예제는 Spring Template Project를 기반으로 Maven을 사용하여 작성 하였다. 실제 사용자가 추가하는 library는 DB 접속 라이브러리(Altibase.jar), Mybatis 관련 라이브러리(mybatis-3.2.8.jar), Mybatis-Spring 연동 라이브러리(Mybatis-spring-1.2.1.jar)만 필요하며 Spring 관련 라이브러리는 Maven을 이용하여 처리한다.(Spring Template Project에서 Spring MVC Project를 통해 작성하였으며 해당 모드로 작성시 생성되는 pom.xml 파일에 dependency 태그를 이용하여 자동으로 Library를 import 한다. 자세한 내용은 부록에 명시 한다.)



Connection Pool 설정 방법

Spring-Mybatis가 연동됨에 따라 `dataSource`를 설정하는 부분이 독립적으로 설정이 가능하게 된다. Spring 연동 없이 Mybatis에서만 다른 모듈의 Connection Pool를 사용할 수 없으며 Spring과 연동을 해야 다른 모듈의 Connection Pool를 사용할 수 있다는 것을 주의해야 한다.

아래 내역은 Connection Pool을 설정 하는 예제 이다.

예) altibase_mybatis_spring의 applicationContext.xml 파일

```
...
<!-- Altibase Connection Pool-->
<bean id="dataSource" class="Altibase.jdbc.driver.AltibaseConnectionPoolDataSource">
    <property name="URL" value="jdbc:Altibase://192.168.1.62:21020/mydb" />
    <property name="user" value="test" />
    <property name="password" value="test" />
</bean>
...

<!-- Apache Connection Pool-->
<bean id="dataSource"
class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
<property name="driverClassName" value="Altibase.jdbc.driver.AltibaseDriver" />
<property name="url" value="jdbc:Altibase://192.168.1.35:36492/mydb" />
<property name="username" value="sys" />
<property name="password" value="manager" />
<property name="initialSize" value="10" />
<property name="minIdle" value="25" />
<property name="maxIdle" value="30" />
<property name="maxActive" value="100" />
<property name="validationQuery" value="select 1 from dual" />
</bean>
```

트랜잭션 관리

MyBatis에서 DB와 연동할 경우 Configuration 파일의 <environments> 의 DB별 <environment> 태그의 <transactionManager>에 dataSource를 지정하고 <environment>의 id를 가지고 SqlSessionFactory 객체를 통해 DB와 연결하며 이 때 auto-commit mode도 설정이 가능하다. Default는 autocommit 모드이다.

본 장에서는 이러한 트랜잭션 관리 방법들을 소개한다.

MyBatis 에서 트랜잭션 관리

MyBatis에서 DB와 연동할 경우 iBatis에서 설정 파일 하나에 한 개의 DB를 설정할 수 있었던 것과 다르게 MyBatis는 DB 설정을 <environments> 태그로 여러 개의 DB를 설정할 수 있게 관리 한다.

MyBatis는 <environments> 내의 <environment> 태그 하나당 하나의 DB를 설정할 수 있으며 <environment> 태그의 <transactionManager>에 dataSource를 지정 하게 되며 어플리케이션에서 프로그래머가 직접 트랜잭션을 관리할 수 있다.

MyBatis에서는 트랜잭션 시작과 끝은 따로 존재하지 않으며 Configuration을 통해 SqlSessionFactory를 설정할 때 AutoCommit 모드 설정이 가능하다.

다음의 예제는 다중 DB를 하나의 Configuration 파일로 설정하는 예제이다.

예) altibase_mybatis_MultiVersionConnection의 mybatis-config.xml 파일

```
...
<!-- DB 연결 옵션 : UNPOOLED/POOLED/JNDI -->
<!-- transactionManager 옵션 : JDBC/MANAGED -->
<environments default="development">
<!-- development alias의 connection configuration(5 version) -->
<environment id="development">
  <transactionManager type="JDBC" />
  <dataSource type="POOLED">
    <property name="driver" value="${jdbc5.driver}" />
    <property name="url" value="${jdbc5.url}" />
    <property name="username" value="${jdbc5.username}" />
    <property name="password" value="${jdbc5.password}" />
    <property name="poolPingQuery" value="select 1 from dual" />
    <property name="poolMaximumActiveConnections" value="100" />
    <property name="poolMaximumIdleConnections" value="50" />
    <property name="poolMaximumCheckoutTime" value="20000" />
  </dataSource>
</environment>
<!-- release alias의 connection configuration(6 or later version) -->
<environment id="release">
  <transactionManager type="JDBC" />
  <dataSource type="POOLED">
    <property name="driver" value="${jdbc.driver}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
    <property name="poolPingQuery" value="select 1 from dual" />
    <property name="poolMaximumActiveConnections" value="100" />
    <property name="poolMaximumIdleConnections" value="50" />
    <property name="poolMaximumCheckoutTime" value="20000" />
  </dataSource>
</environment>
</environments>
```

```
</environment>
</environments>
...
```

예) altibase_mybatis_MultiVersionConnection의 MybatisUtil.java 파일

```
...
// mybatis-config.xml로부터 설정값을 읽어옴
InputStream = Resources.getResourceAsStream("mybatis-config.xml");
// 읽어온 xml 파일로부터 SqlSessionFactory를 생성함.(이 때 environment id를
// nel에 인자값으로 부여하여 연결하려는 DB 설정
sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream, "release");
// sqlSession 객체 생성. 이 때 인자값이 autoCommit 여부를 결정함.
// default는 true이며 false 값을 주게 되면 autocommit=false
sqlSession = sqlSession.openSession(false);
...
```

위의 예제 altibase_mybatis_MultiVersionConnection 프로젝트를 실행하기 위해서는
"Configuration 파일에 dataSource를 설정하여 ALTIBASE와 연동"과 마찬가지로
Altibase.jar, mybatis-3.2.8.jar 파일이 필요하다.

MyBatis 연동 시 고려사항

Spring에서 ALTIBASE에 연동할 경우 고려해야 할 사항에 대해 설명한다.

LOB 데이터 처리

iBatis에서는 LOB을 처리하기 위해서는 Mapper 파일에 parameter와 result에 대한 정보를 setting하는 부분에 반드시 jdbcType을 CLOB/BLOB이라고 명시해줘야 하며 parameterMap을 이용하여 처리 하였지만, Mybatis에서는 parameterMap이 deprecated 되었기 때문에 쿼리의 binding하는 컬럼에 jdbcType을 직접 명시 한다. 명시하지 않을 경우 LOB 데이터로 인식하지 않아 에러를 발생 하게 된다.

다음은 CLOB 타입의 데이터에 대한 parameterMap과 resultMap을 지정하여 setting하고 있는 예제이다.

예) altibase_mybatis_lob의 LobMapper.xml(Mapper) 파일

```
<mapper namespace="com.altibase.psy.mapper.LobMapper">
<!-- LOB select, insert, update, delete Test - 2013/11/05 -->
<select id="selectBlobData" parameterType="Integer" resultType="LobVo">
    SELECT user_no as userNo, user_name as userName, blob_data as blobData,
    clob_data as clobData, reg_date as regDate
    FROM test_blob
    WHERE user_no = #{userNo}
</select>

<select id="selectAllBlobData" resultType="LobVo">
    SELECT user_no as userNo, user_name as userName, blob_data as blobData,
    clob_data as clobData, reg_date as regDate
    FROM test_blob
</select>

<insert id="insertBlobData" parameterType="LobVo">
    insert into test_blob(user_no, user_name, blob_data, clob_data, reg_date)
    values(#{userNo}, #{userName}, #{blobData,jdbcType=BLOB},
    #{clobData,jdbcType=CLOB}, #{regDate})
</insert>

<update id="updateBlobData" parameterType="LobVo">
    update test_blob set
        user_name = #{userName},
        blob_data = #{blobData,jdbcType=BLOB},
        clob_data = #{clobData,jdbcType=CLOB},
        reg_date = #{regDate}
    where user_no = #{userNo}
</update>
...
</mapper>
```

또한 LOB 처리 시 반드시 주의해야 할 사항은 ALTIBASE에서 LOB 데이터를 처리하기 위해서는 반드시 autocommit 모드를 false로 바꾼 후 트랜잭션을 관리해줘야

한다는 것이다. iBATIS 연동 시 SqlMapConfig 파일의 <transactionManager>에 dataSource를 설정할 경우에는 내부적으로 setAutoCommit(false); 메소드를 호출하여 autocommit 모드를 false로 바꾸주기 때문에 LOB 처리 시 따로 고려할 사항은 없으나 MyBatis에서는 default가 true이기 때문에 Session을 얻어올 때 명시해 주어야 한다.

만약 autocommit 모드를 false로 변경해주지 않으면, "java.sql.SQLException: [0]:LobLocator can not span the transaction 101858625." 과 같은 예러가 발생한다.

그리고 LOB 데이터를 입력 시에도 "java.sql.SQLException: [0]:LobLocator can not span the transaction 101858625." 예러가 발생하게 된다.

다음의 예제는 MyBatis에서 Lob 데이터를 처리하는 Sample의 일부이며 전체 소스는 부록에 첨부 하였다.

예) altibase_mybatis_lob의 LobMapper.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.altibase.psy.mapper.LobMapper">
  <!-- LOB select, insert, update, delete Test - 2013/11/05 -->
  <select id="selectBlobData" parameterType="Integer" resultType="LobVo">
    SELECT user_no as userNo, user_name as userName, blob_data as
blobData, clob_data as clobData, reg_date as regDate
    FROM test_blob
    WHERE user_no = #{userNo}
  </select>

  <select id="selectAllBlobData" resultType="LobVo">
    SELECT user_no as userNo, user_name as userName, blob_data as
blobData, clob_data as clobData, reg_date as regDate
    FROM test_blob
  </select>

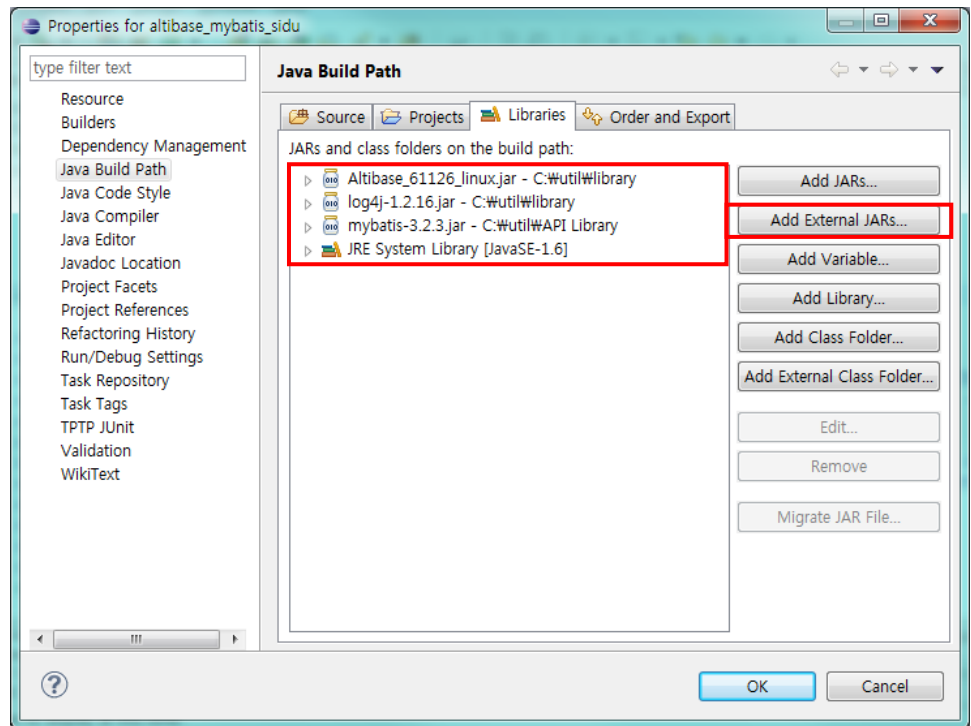
  <insert id="insertBlobData" parameterType="LobVo">
    insert into test_blob(user_no, user_name, blob_data, clob_data,
reg_date)
    values(#{userNo}, #{userName}, #{blobData,jdbcType=BLOB},
#{clobData,jdbcType=CLOB}, #{regDate})
  </insert>

  <update id="updateBlobData" parameterType="LobVo">
    update test_blob set
      user_name = #{userName},
      blob_data = #{blobData,jdbcType=BLOB},
      clob_data = #{clobData,jdbcType=CLOB},
      reg_date = #{regDate}
    where user_no = #{userNo}
  </update>

  <delete id="deleteBlobData" parameterType="Integer">
    delete from test_blob
    where user_no = #{userNo}
  </delete>

</mapper>
```

위의 altibase_mybatis_lob 프로젝트를 실행하기 위해서는 Altibase.jar, mybatis-3.2.8.jar 파일이 필요 하다.



Insert시 insert query가 중복되어 보내지는 현상

```

15:15:55.076 [main] [main] [ERROR] - 2. Connection.prepareStatement(insert into TB_DB_CONFIGSET_LEVEL(DB_TYPE_CODE, LEVEL_CODE, LEVEL_NAME
)
values (?, ?, ?
), 1) insert into TB_DB_CONFIGSET_LEVEL(DB_TYPE_CODE, LEVEL_CODE, LEVEL_NAME
)
values (?, ?, ?
)

```

문제가 나오는 현상 :

- myBatis 설정 xml 내에서 'useGeneratedKeys' 설정을 true로 하면 위의 문제점 발생
- 위 설정은 아래와 같이 정의되어 있음 (<http://mybatis.github.io/mybatis-3/ko/sqlmap-xml.html>)
- 데이터베이스에서 내부적으로 생성한 키를 받는 JDBC getGeneratedKeys 메시지를 사용하도록 설정한다.
- default value : 'false'
- 이에 해당 설정을 false로 변경 후 발생하지 않음
- 기 설정은 Oracle과 informix 내에서는 오류가 발생하지 않음

부록(Mybatis-Altibase 연동)

Altibase_mybatis_sidu 예제를 바탕으로 MyBatis에서 ALTIBASE와 연동하는 방법에 대해 좀 더 자세하게 설명한다. 단, IDE는 Eclipse를 사용한다.

DB 테이블 및 시퀀스 생성

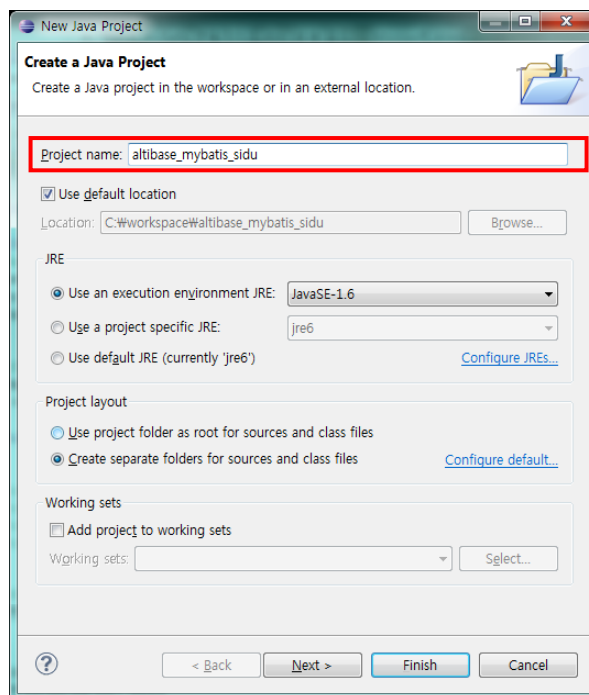
DB에 다음의 테이블과 시퀀스를 생성한다.

```
create table users(  
  user_no integer,  
  user_name varchar(30),  
  user_content varchar(200),  
  reg_date date  
);
```

프로젝트 생성

Eclipse에서 altibase_mybatis_sidu 이라는 프로젝트를 생성한다.

1. 메뉴 - File - Java Project 클릭
2. Project name : 에 Altibase_mybatis_sidu 입력
3. Finish 버튼을 클릭



패키지 명명 규칙

부록에서 설명하고 있는 패키지 명은 기본적으로 `com.altibase.sidu`를 명명하여 사용하고 있는 데 이는 일반적으로 사용하는 패키지 분류 법칙을 가져와 적용한 것이며 각각의 성격은 다음과 같다.

- `com`: 첫번째 항목은 프로젝트를 이끄는 그룹의 성격을 결정하는 것으로 `com`은 `company`를 의미 한다. 만약 소규모 단체 등이라면 `org(organization)`를 사용할 것이다.
- `altibase`: 두번째 항목은 자사의 그룹 또는 사명을 정해주는 부분으로 보통 회사라면 회사명이, 특정 그룹이라면 그룹명이 들어간다. 현 문서는 ALTIBASE에서 작성 하였으므로 `altibase`로 명명 하였다.
- `sidu`: 세 번째 항목은 실제 이 프로젝트의 `artifact` 구조를 의미 한다. 현재 Sample은 DML의 일반적인 명령(`Select/Insert/Delete/Update`)이므로 `sidu`로 명명 하였다.

Configuration 파일 작성

1. ALTIBASE 연결을 위한 `property`들을 정의한 `properties` 파일(`db.properties`)을 작성한다. (`altibase_mybatis_sidu` 프로젝트 - `src` 디렉토리에서 마우스 오른쪽 버튼 클릭하여 `New - File`을 클릭한다. `File name:` 에 `db.properties`을 작성한다.)

```
jdbc.driver=Altibase.jdbc.driver.AltibaseDriver
jdbc.url=jdbc:Altibase://192.168.1.62:21020/mydb
jdbc.username=sys
jdbc.password=manager
```

2. Configuration 파일(`mybatis-config.xml`)에 ALTIBASE와 연동을 위한 `dataSource`와 `SqlMap` 파일을 설정한다. (`altibase_mybatis_sidu` 프로젝트 - `src` 디렉토리에서 마우스 오른쪽 버튼 클릭하여 `New - File`을 클릭한다. `File name:` 에 `mybatis-config.xml`을 작성한다.)

```
<?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>

<typeAliases>
  <typeAlias type="com.altibase.sidu.model.UserVo" alias="User" />
</typeAliases>

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



```

        <property name="poolPingQuery" value="select 1 from
dual" />
    </dataSource>
</environment>
</environments>

<mappers>
    <mapper resource="com/altibase/sidu/mapper/UserMapper.xml" />
</mappers>

</configuration>

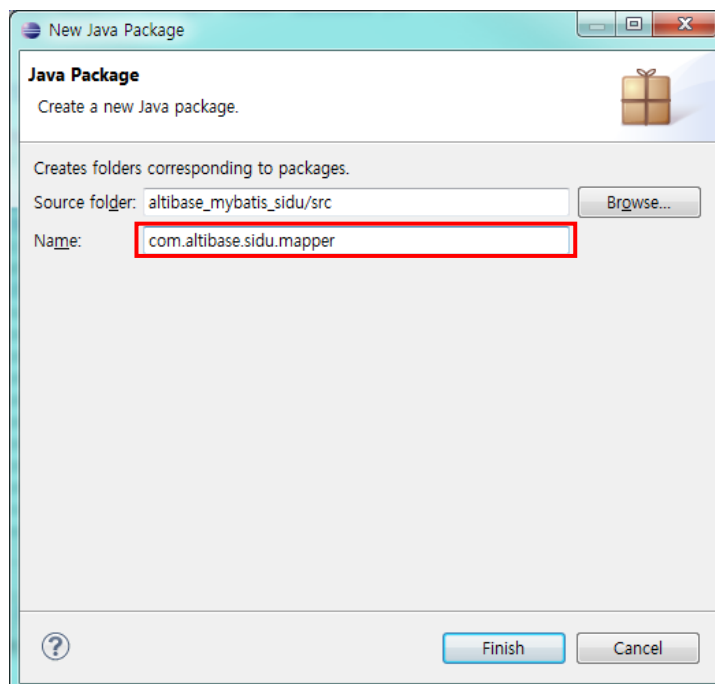
```

Mapper 파일 작성

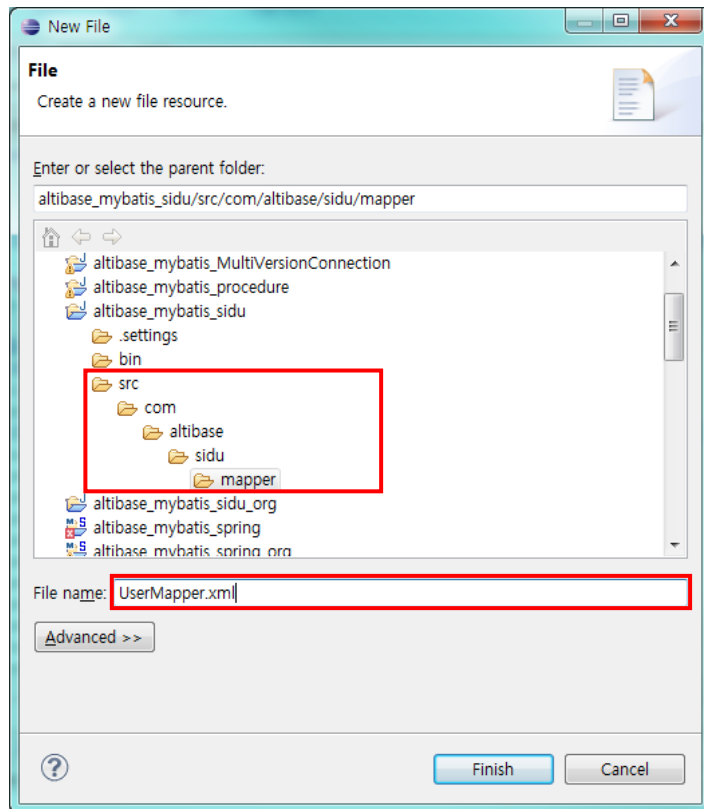
Users 테이블의 CRUD SQL 구문과 mapping되는 method들을 정의한 Mapper 파일을 작성한다.(Person.xml)

Mapper는 com.altibase.sidu.mapper Package에 작성 하게 된다.

1. altibase_mybatis_sidu 프로젝트 - src 디렉토리에서 마우스 오른쪽 버튼 클릭후 New - Package을 클릭하여 com.altibase.sidu.mapper 라는 신규 패키지를 생성한다.



2. 생성된 신규 패키지(com.altibase.sidu.mapper)에서 마우스 오른쪽 버튼 클릭 후 New - File을 클릭하여 File name: 에 UserMapper.xml을 작성한다.



3. 다음의 내용을 UserMapper.xml에 작성 한다

```

<?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.altibase.sidu.mapper.UserMapper">
  <select id="selectUserData" parameterType="Integer" resultType="User">
    SELECT user_no as userNo,
    user_name as userName,
    user_content as userContent,
    reg_date as regDate
    FROM users
    WHERE user_no = #{userNo}
  </select>

  <select id="selectAllUserData" resultType="User">
    SELECT user_no as userNo,
    user_name as userName,
    user_content as userContent,
    reg_date as regDate
    FROM users
  </select>

  <insert id="insertUserData" parameterType="User">
    insert into
    users(user_no, user_name, user_content, reg_date)
    values(#{userNo}, #{userName}, #{userContent}, #{regDate})
  </insert>

  <update id="updateUserData" parameterType="User">

```

```

update users
set user_name = #{userName},
user_content = #{userContent},
reg_date = #{regDate}
where user_no = #{userNo}
</update>

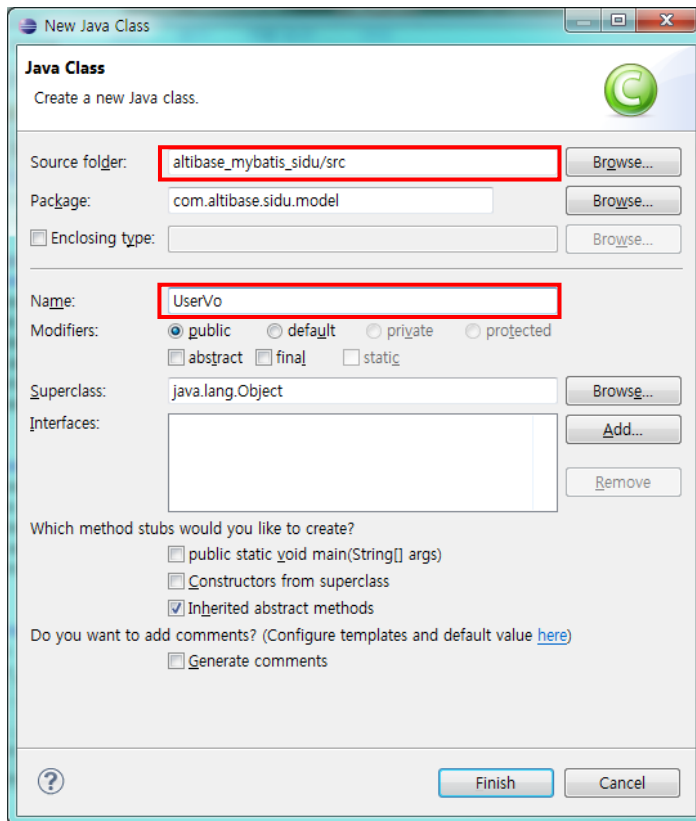
<delete id="deleteUserData" parameterType="User">
delete from users
where user_no = #{userNo}
</delete>
</mapper>

```

Application에서 Configuration의 insert, update, delete Method를 호출할 때 위의 파일에 정의되어 있는 <insert>, <update>, <delete>, <select> 태그에 정의되어 있는 id와 일치하는 SQL 문들이 자동으로 수행이 된다.

Application 작성

1. users 테이블에 대한 VO(Value Object) 객체인 UserVO 클래스(UserVo.java)를 작성한다.
 - 1-1. altibase_mybatis_sidu 프로젝트 - src 디렉토리에서 마우스 오른쪽 버튼 클릭 후 New - Class를 클릭 한다.
 - 1-2. Package:에 com.altibase.sidu.model를 입력하고 Name: 에 UserVo를 입력한다.



1-3. 다음의 내용을 UserVo.java 파일에 작성 한다.

```
package com.altibase.sidu.model;

import java.io.Serializable;
import java.util.Date;

@SuppressWarnings("serial")
public class UserVo implements Serializable {
    private Integer userNo;
    private String userName;
    private String userContent;
    private Date regDate;

    public Integer getUserNo() {
        return userNo;
    }

    public void setUserNo(Integer userNo) {
        this.userNo = userNo;
    }

    public String getUserName() {
        return userName;
    }

    public void setName(String userName) {
        this.userName = userName;
    }

    public String getUserContent() {
        return userContent;
    }

    public void setUserContent(String userContent) {
        this.userContent = userContent;
    }

    public Date getRegDate() {
        return regDate;
    }

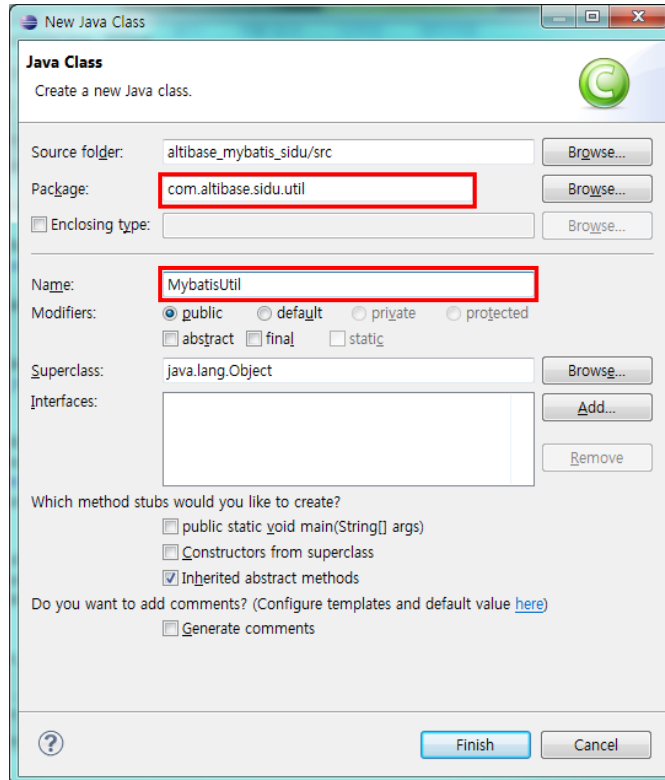
    public void setRegDate(Date regDate) {
        this.regDate = regDate;
    }

    public String toString() {
        return "user_no = " + this.userNo + "\n" + "user_name = "
            + this.userName + "\n" + "user_content = " +
this.userContent
            + "\n" + "reg_date = " + this.regDate;
    }
}
```

2. DB 연결 및 필요한 메소드를 포함하는 MyBatisUtil.java 클래스를 작성 한다.

2-1. altibase_mybatis_sidu 프로젝트 - src 디렉토리에서 마우스 오른쪽 버튼 클릭 후 New - Class를 클릭 한다.

2-2. Package:에 com.altibase.sidu.util을 입력하고 Name: 에 MybatisUtil을 입력 한다.



2-3. 다음의 내용을 MybatisUtil.java 파일에 작성 한다

```
package com.altibase.sidu.util;

import java.io.IOException;
import java.io.InputStream;
import java.text.ParsePosition;
import java.text.SimpleDateFormat;
import java.util.Calendar;

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

public class MybatisUtil {

    private static SqlSessionFactory sqlSessionFactory;
    private static SqlSession sqlSession;

    public static SqlSession getSqlSessionFactory() {
        if (sqlSessionFactory == null) {
            InputStream inputStream;
            try {
                inputStream = Resources
                    .getResourceAsStream("mybatis-
```

```

config.xml");
        sqlSessionFactory = new
SqlSessionFactoryBuilder().build(inputStream);
        sqlSession = sqlSessionFactory.openSession(false);
    } catch (IOException e) {
        throw new RuntimeException(e.getCause());
    }
}
return sqlSession;
}

public String NullToStr(String str) {
    if (str == null) {
        str = "";
    }
    return str;
}

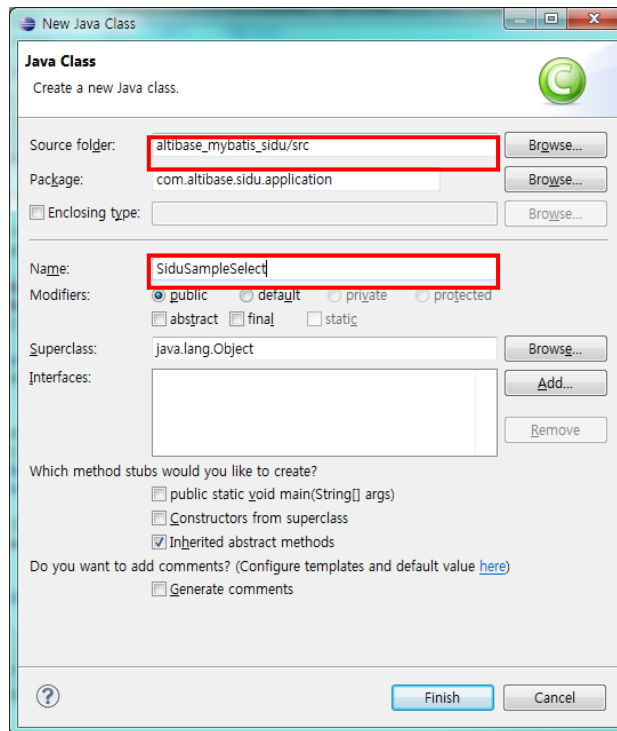
public String getFormatDate(String date, String originalformat,
    String wantformat) {
    java.util.Date d = null;
    SimpleDateFormat dd = new SimpleDateFormat(originalformat);
    ParsePosition parse = new ParsePosition(0);
    d = dd.parse(date, parse);
    Calendar cal = Calendar.getInstance();
    cal.setTime(d);
    SimpleDateFormat sdf = new SimpleDateFormat(wantformat);
    String day = sdf.format(cal.getTime());
    return day;
}
}

```

3. MybatisUtil 클래스로부터 sqlSession 객체를 받아 쿼리를 수행하는 클래스를 작성한다. 예제에서는 DML 각각의 처리에 대한 클래스를 별도로 선언하여 처리하였다. 해당 예제에서는 Select에 대한 예시만 설명한다.

3.1 altibase_mybatis_sidu 프로젝트 - src 디렉토리에서 마우스 오른쪽 버튼 클릭 후 New - Class를 클릭 한다.

3.2 Package:에 com.altibase.sidu.application을 입력하고 Name: 에 SiduSampleSelect를 입력 한다.



3.3 다음의 내용을 SiduSampleSelect.java 파일에 작성 한다.

```

package com.altibase.sidu.application;

import java.sql.SQLException;
import java.util.List;
import com.altibase.sidu.model.UserVo;
import com.altibase.sidu.util.MybatisUtil;

public class SiduSampleSelect {

    public UserVo SiduSelect(int user_no) {
        UserVo userVo = new UserVo();

        try {
            String statement =
"com.altibase.sidu.mapper.UserMapper.selectUserData";
            userVo =
MybatisUtil.getSqlSessionFactory().selectOne(statement,
                user_no);

            return userVo;
        } catch (Exception e) {
            throw new IllegalArgumentException(e);
        }
    }

    public List<UserVo> SiduSelectList() throws SQLException {
        String statement =
"com.altibase.sidu.mapper.UserMapper.selectAllUserData";
        List<UserVo> userVos =
MybatisUtil.getSqlSessionFactory().selectList(statement);

        return userVos;
    }
}

```

4. 실제 프로그램이 수행되는 Main 클래스를 SiduMain.java 파일에 작성 한다.

```
package com.altibase.sidu.application;

import java.util.Calendar;
import java.util.List;

import com.altibase.sidu.model.UserVo;

public class SiduMain {

    /**
     * @param args
     */
    public static void main(String[] args) {

        UserVo userVo = new UserVo();
        int user_no = 1;
        int res_count = 0;

        try {

            // Insert 예제
            userVo.setUserNo(user_no);
            userVo.setUserName("sidu_1");
            userVo.setUserContent("sidu_content_1");
            userVo.setRegDate(Calendar.getInstance().getTime());
            SiduSampleInsert sidu_insert = new
SiduSampleInsert(userVo);
            res_count = sidu_insert.SiduInsert();

            // Select 예제
            SiduSampleSelect sidu_select = new SiduSampleSelect();
            userVo = sidu_select.SiduSelect(user_no);

            System.out.println(userVo.toString());

            // Insert 예제
            userVo.setUserNo(5);
            sidu_insert = new SiduSampleInsert(userVo);
            res_count = sidu_insert.SiduInsert();
            System.out.println("insert count = " + res_count);

            // Update 예제
            userVo.setUserName("sidu_5");
            userVo.setUserContent("sidu_content_5");
            userVo.setRegDate(Calendar.getInstance().getTime());
            SiduSampleUpdate sidu_update = new
SiduSampleUpdate(userVo);
            res_count = sidu_update.SiduUpdate();
            System.out.println("update count = " + res_count);

            // Delete 예제
            user_no = 5;
            SiduSampleDelete sidu_delete = new
SiduSampleDelete(user_no);
            res_count = sidu_delete.SiduDelete();
            System.out.println("delete count = " + res_count);
```



```

// Select All 예제
List<UserVo> userVos = sidu_select.SiduSelectList();

for(int i=0; i<userVos.size(); i++) {
    userVo = userVos.get(i);

    System.out.println((i+1) + "번째 객체 ");
    System.out.println(userVo.toString());
}

// 동일한 테스트를 두 번 반복 했을 때, user_no가
1인 데이터가 2개가 되어 select만 수행되고
// 프로그램이 Terminate 될 수 있어서(SelectOne
메소드 사용) user_no가 1인 데이터도 삭제해 준다.
user_no = 1;
sidu_delete = new SiduSampleDelete(user_no);
res_count = sidu_delete.SiduDelete();

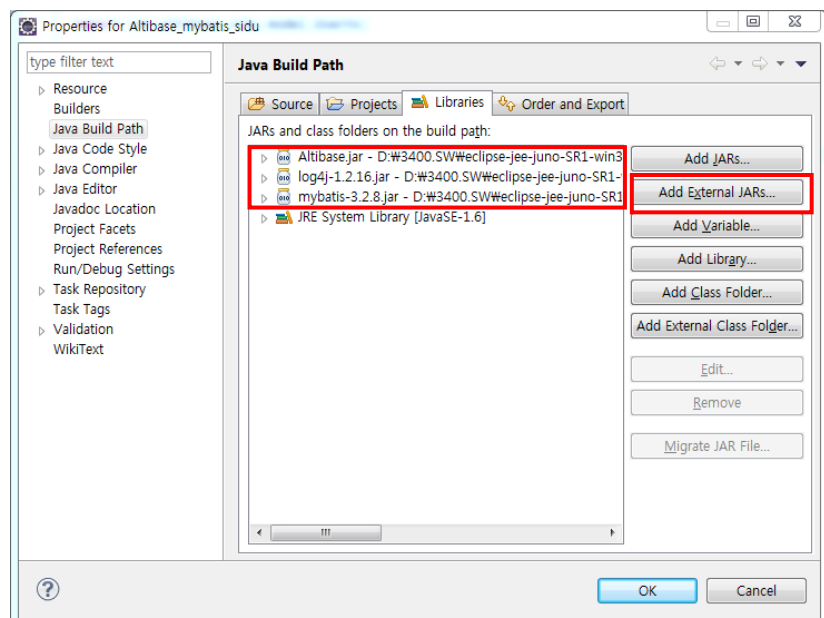
}
catch(Exception e) {
    e.printStackTrace();
}
}
}

```

관련 JAR 파일 추가

Altibase.jar와 mybatis-3.2.8.jar 파일을 추가한다.

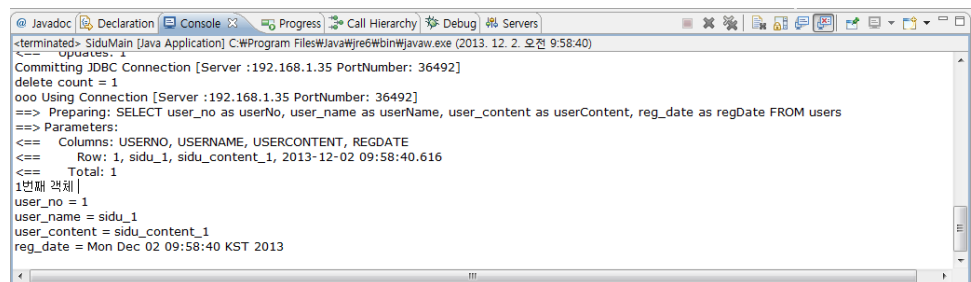
altibase_mybatis_sidu 프로젝트에서 마우스 오른쪽 버튼 클릭하여 Properties를 클릭 - Java Build Path - Libraries 에서 Add External JARS를 클릭하여 Altibase.jar와 mybatis-3.2.8.jar 파일을 추가한다.



Application 실행

altibase_mybatis_sidu 프로젝트를 실행한다.

altibase_mybatis_sidu 프로젝트를 클릭한 후 메뉴에서 Run을 실행하거나 com.altibase.sidu.application 패키지의 SiduMain.java 클래스를 더블 클릭하여 파일을 연 후에 Run을 실행 한다.



```
@ Javadoc | Declaration | Console | Progress | Call Hierarchy | Debug | Servers
<terminated> SiduMain [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2013. 12. 2 오전 9:58:40)
<--- Updates: 1
Committing JDBC Connection [Server :192.168.1.35 PortNumber: 36492]
delete count = 1
ooo Using Connection [Server :192.168.1.35 PortNumber: 36492]
==> Preparing: SELECT user_no as userNo, user_name as userName, user_content as userContent, reg_date as regDate FROM users
==> Parameters:
<== Columns: USERNO, USERNAME, USERCONTENT, REGDATE
<== Row: 1, sidu_1, sidu_content_1, 2013-12-02 09:58:40.616
<== Total: 1
1번째 객체 |
user_no = 1
user_name = sidu_1
user_content = sidu_content_1
reg_date = Mon Dec 02 09:58:40 KST 2013
```

부록 2(Spring-Mybatis-Altibase 연동)

altibase_mybatis_spring 예제를 바탕으로 Spring-MyBatis-ALTIBASE와 연동하는 방법에 대해 좀 더 자세하게 설명한다.

단, IDE는 Eclipse를 사용한다.

DB 테이블 및 시퀀스 생성

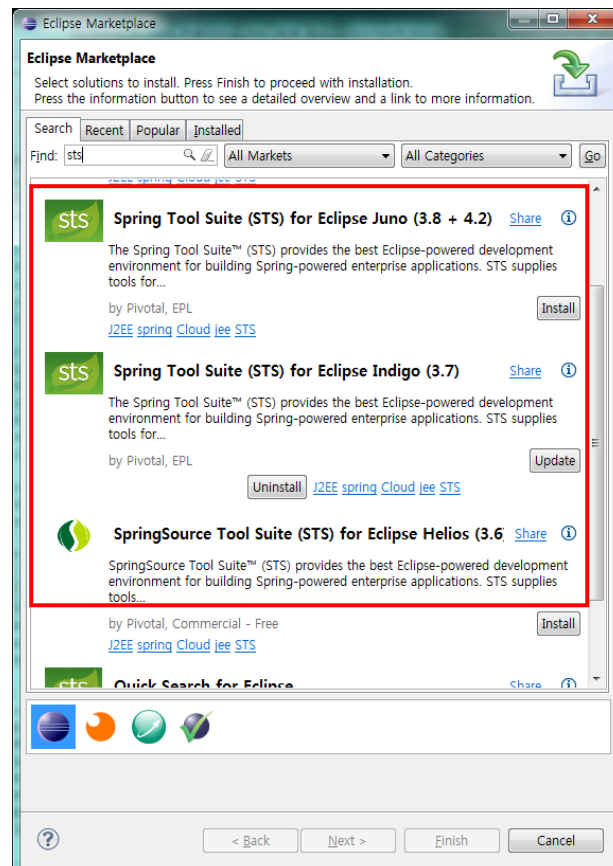
DB에 다음의 테이블과 시퀀스를 생성한다.

```
create table users(  
  user_no integer,  
  user_name varchar(30),  
  user_content varchar(200),  
  reg_date date  
);
```

Spring 설치

Spring을 사용하려면 STS(STS, SpringSource Tools Suite)를 설치해야 한다.

STS란 이클립스에서 Spring으로 개발시에 편리하게 클립에서 스프링으로 개발할때 편리를 제공하는 플러그인으로 아래 목록 중 Eclipse의 Version에 맞게 설치 한다.



Maven 설치

- Maven은 Apache Software Foundation에서 개발되고 있는 소프트웨어 프로젝트 관리 툴이다.

Maven은 Project Object Model(POM)이라는 것에 기초를 두어 프로젝트의 빌드, 테스트, 문서화 등 프로젝트 라이프사이클 전체를 관리할 수 있는 툴이며 Spring-Mybatis-Altibase 연동 테스트 소스도 Maven을 적용 하여 관리하게 되었다.

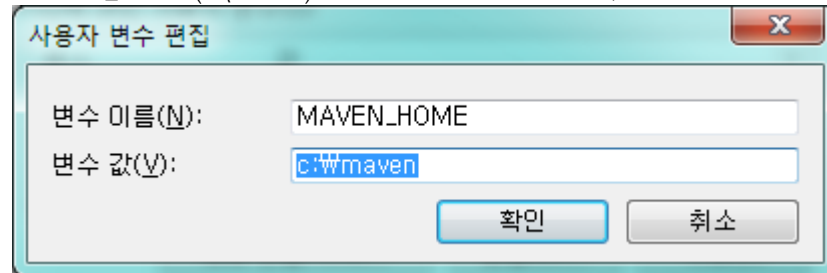
해당 part는 Maven의 설치 과정을 설명한다.

1. Maven 파일 다운로드

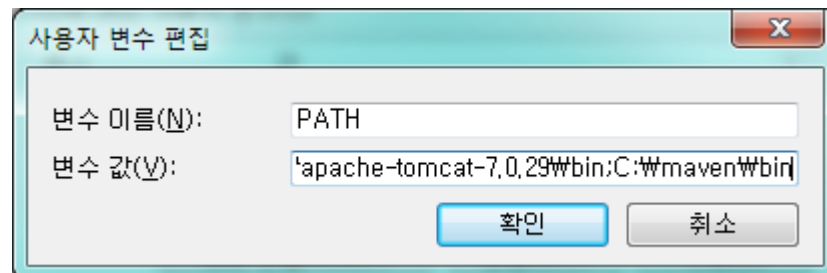
- <http://maven.apache.org/download.html> 에서 binary zip 파일을 다운 받는다.
2014년 10월 현재 최신버전은 [apache-maven-3.2.8-bin.zip](#)

2. 메이븐 환경 설정

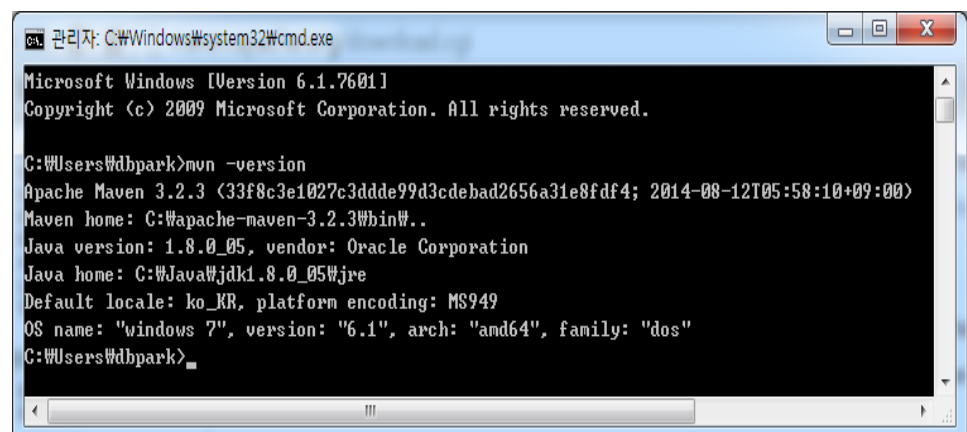
- 압축을 해제한 후에 윈도우 환경변수(바탕화면->내컴퓨터->속성->고급->환경변수)에서 MAVEN_HOME(c:\maven)을 사용자 변수에 추가하고,



PATH 설정에 maven binary path를 포함 시킨다.

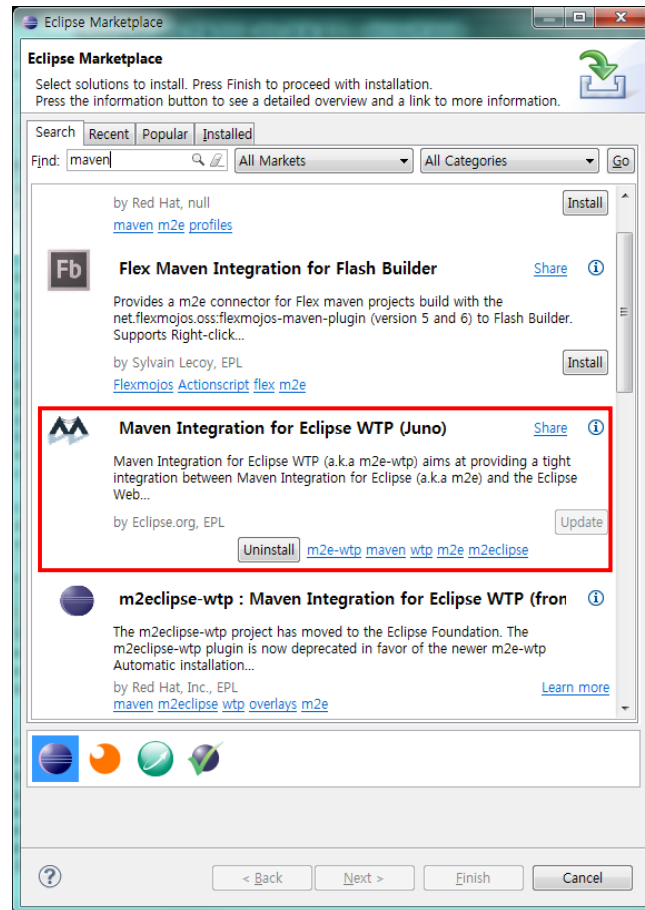


- 윈도우에서 시작 -> 실행 -> cmd 입력 후 도스 창 뜨면 mvn -version 입력하여 버전이 뜨면 OK



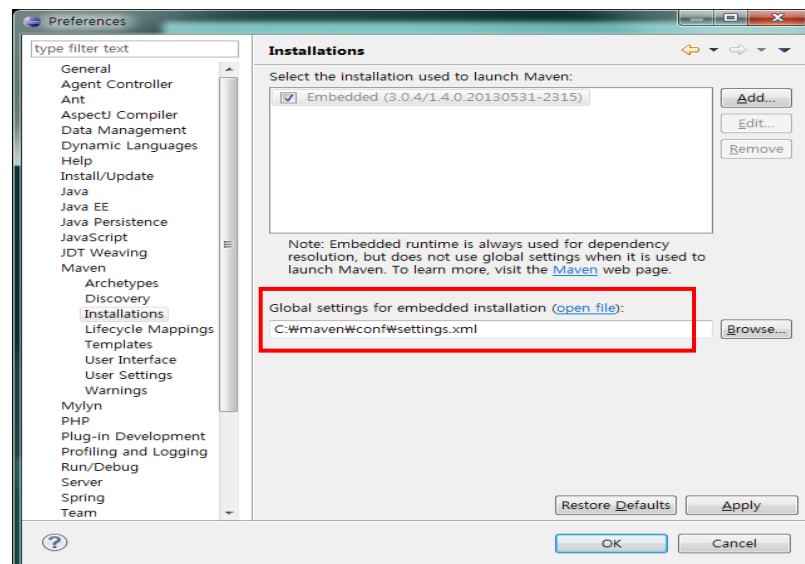
3. eclipse에서 maven 다운로드

- eclipse의 Help->Eclipse Marketplace를 클릭하여 STS를 다운받는 것과 같이 maven을 다운로드 받는다.



4. 압축을 해제한 maven을 eclipse에 설정 한다.

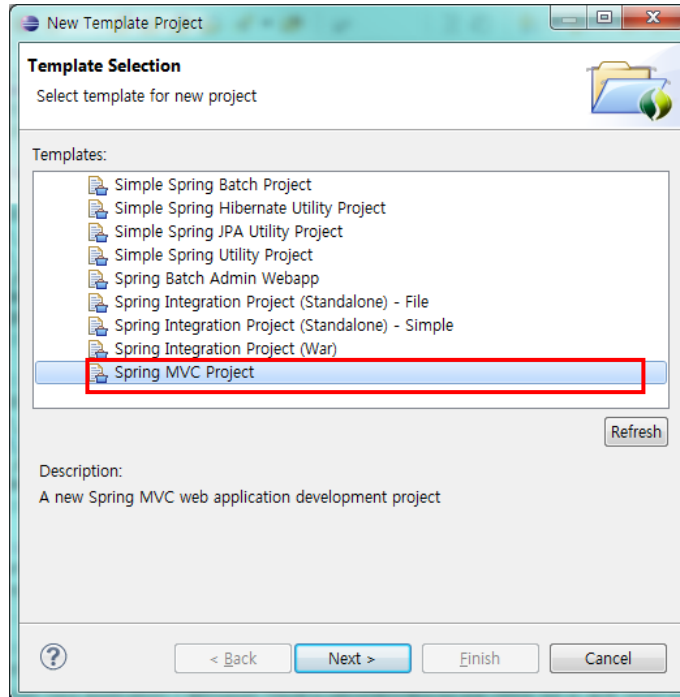
- windows -> preference -> Maven -> Installatio 아래 browse클릭하여 "C:\maven\conf\settings.xml" 경로를 잡아주고 Apply.



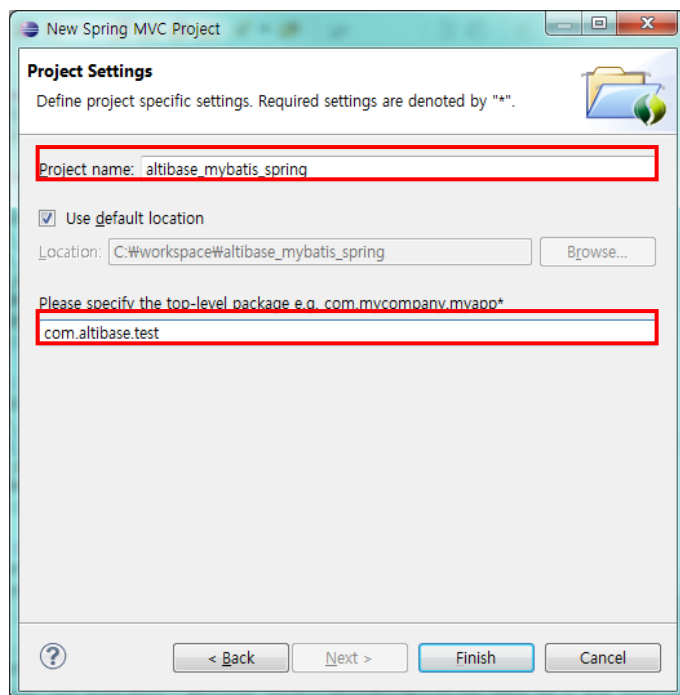
프로젝트 생성

Eclipse에 STS를 설치하면 Spring Template Project의 Spring 기반 프로젝트가 생성이 가능하다. 다음의 단계를 거쳐 프로젝트를 생성 한다.

1. 메뉴-File-New-Spring Template Project를 클릭하여 Spring MVC Project를 생성



2. Project name : 예 altibase_mybatis_spring, top-level package에 com.altibase.test 입력(Spring은 프로젝트 생성시 default 패키지 명을 필수로 입력해야 한다.)



3. Finish 버튼을 클릭

패키지 명명 규칙

부록에서 설명하고 있는 패키지 명은 기본적으로 `com.altibase.test`를 명명하여 사용하고 있는 데 이는 일반적으로 사용하는 패키지 분류 법칙을 가져와 적용한 것이며 각각의 성격은 다음과 같다.

- `com`: 첫번째 항목은 프로젝트를 이끄는 그룹의 성격을 결정하는 것으로 `com`은 `company`를 의미 한다. 만약 소규모 단체 등이라면 `org(organization)`를 사용할 것이다.
- `altibase`: 두번째 항목은 자사의 그룹 또는 사명을 정해주는 부분으로 보통 회사라면 회사명이, 특정 그룹이라면 그룹명이 들어간다. 현 문서는 ALTIBASE에서 작성 하였으므로 `altibase`로 명명 하였다.
- `test`: 세 번째 항목은 실제 이 프로젝트의 artifact 구조를 의미 한다. 현재 Samle은 DML의 일반적인 명령(Select/Insert/Delete/Update)이므로 `test`로 명명 하였다.

ApplicationContext 파일 작성

스프링 코드 작성에서 가장 먼저 해야 할 것은 `AppilcationContext`라고 불리는 스프링 빈 설정 파일을 작성 하는 것이다. 아래는 스프링 빈 설정 파일의 내역이다.

```
<?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:tx="http://www.springframework.org/schema/tx"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-3.1.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd">

    <description>Example configuration to get you started.</description>

    <!-- 지정된 패키지에서 빈을 탐색 -->
    <context:component-scan base-package="com.altibase.test.service" />

    <!-- Apache DBCP를 이용한 ALTIBASE DB 접속 방법 - Apache
    DBCP라 초기에 Session Pool을 붙이는 프로퍼티가 존재하며
    물론 최초 연결시에 initialSize만큼 Session을 만들어 두는 것도
    가능 하다. -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
    destroy-method="close">
        <property name="driverClassName"
value="Altibase.jdbc.driver.AltibaseDriver" />
        <property name="url" value="jdbc:Altibase://192.168.1.62:21020/mydb" />
        <property name="username" value="test" />
        <property name="password" value="test" />
        <property name="initialSize" value="10" />
        <property name="minIdle" value="25" />
        <property name="maxIdle" value="30" />
        <property name="maxActive" value="100" />
    </bean>
</beans>
```

```

        <property name="validationQuery" value="select 1 from dual" />
        <property name="testOnBorrow" value="false" />
    </bean>

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

    <!-- 트랜잭션 annotation 정의 -->
    <tx:annotation-driven transaction-manager="txManager" />

    <!-- SqlSessionFactory 정의 -->
    <!-- typeAliasesPackage 속성에는 매퍼 정의 파일(XML) 내의 SQL
문장에서 참조하는 도메인 클래스가 포함된 패키지를 지정 -->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="typeAliasesPackage"
value="com.altibase.test.domain" />
    </bean>

    <!-- 매퍼 인터페이스와 매퍼 정의 파일에 의해 생성된 매퍼 클래스를
빈에 주입 할 수 있도록 함 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.altibase.test.persistence"
/>
    </bean>
</beans>

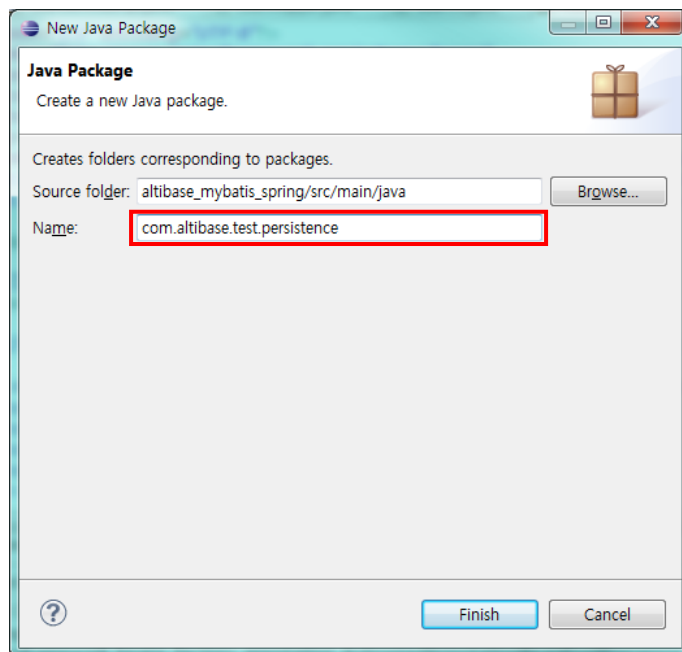
```


Mapper 파일 작성

Users 테이블의 CRUD SQL 구문과 mapping되는 method들을 정의한 Mapper 파일을 작성한다. Spring의 방식을 따라야 하기 때문에 의존적 주입(Dependency Injection) 방식을 사용하게 된다.(UserMapper.xml, UserMapper.java)

Mapper는 com.altibase.test.persistence Package에 작성 하게 된다.

1. altibase_mybatis_spring 프로젝트 - src 디렉토리에서 마우스 오른쪽 버튼 클릭후 New - Package을 클릭하여 com.altibase.test.persistence 라는 신규 패키지를 생성한다.



2. 생성했던 신규 패키지(com.altibase.test.persistence)에 마우스 오른쪽 버튼 클릭 후 New - Interface를 클릭하여 UserMapper.java 파일을 생성한다
3. 생성한 Interface 파일의 내용을 다음과 같이 작성 한다.

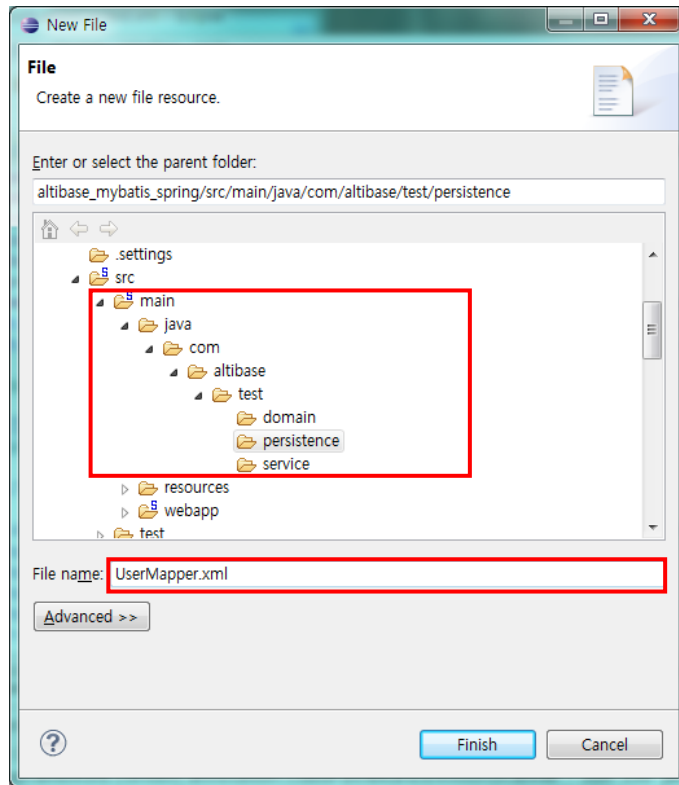
```
package com.altibase.test.persistence;

import java.util.List;

import com.altibase.test.domain.UserVo;

public interface UserMapper {
    public List<UserVo> getAllUser();
    public UserVo getUserByNo(int user_no);
    public Integer insertUserData(UserVo userVo);
    public Integer deleteUserByNo(int user_no);
    public Integer updateUserByNo(UserVo userVo);
}
```

4. 생성된 신규 패키지(`com.altibase.test.persistence`)에서 마우스 오른쪽 버튼 클릭 후 `New - File`을 클릭하여 `File name:` 에 `UserMapper.xml`을 작성한다.



5. 다음의 내용을 `UserMapper.xml`에 작성 한다.

```
<?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.altibase.test.persistence.UserMapper">
    <select id="getAllUser" parameterType="Integer" resultType="UserVo">
        SELECT user_no as userNo,
        user_name as userName,
        user_content as userContent,
        reg_date as regDate
        FROM users
        WHERE user_no = #{userNo}
    </select>

    <select id="getUserByNo" resultType="UserVo">
        SELECT user_no as userNo,
        user_name as userName,
        user_content as userContent,
        reg_date as regDate
        FROM users
    </select>

    <insert id="insertUserData" parameterType="UserVo">
        insert into
        users(user_no, user_name, user_content, reg_date)
        values(#{userNo}, #{userName}, #{userContent}, #{regDate})
    </insert>
</mapper>
```

```
<update id="updateUserByNo" parameterType="UserVo">
    update users
    set user_name = #{userName},
    user_content = #{userContent},
    reg_date = #{regDate}
    where user_no = #{userNo}
</update>

<delete id="deleteUserByNo" parameterType="Integer">
    delete from users
    where user_no = #{userNo}
</delete>
</mapper>
```

해당 방식은 Spring만의 방식으로 의존적 주입(Dependency Injection)의 방식으로 처리하고 있다.

Application에서는 인터페이스에 정의되어 있는 메소드를 통하여 SQL 문을 처리한다.

Dependency Injection

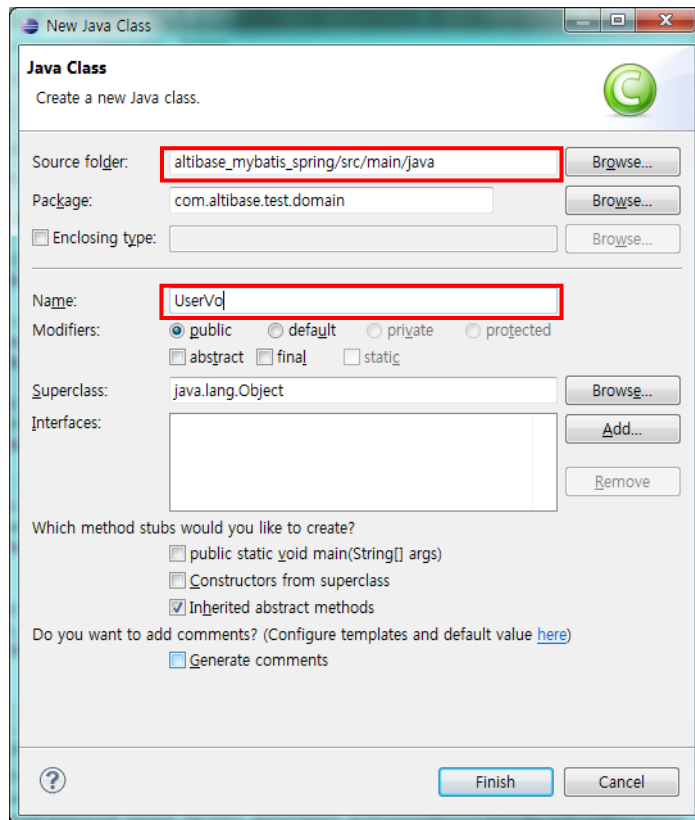
Dependency Injection(줄여서 DI)는 직역하면 의존적 주입 이다.

Dependency Injection이란 Gof의 디자인 패턴에 있는 기법으로, 실제 사용자가 사용하게 될 메소드를 인터페이스로 지정하여 실제 로직과 분리하여 처리하는 기법이다.

이렇게 처리하게 되면 사용자는 인터페이스의 메소드만 이용 하더라도 구현부는 나중에 주입을 통해 해결하므로 획기적인 분업과 동시에 구현 클래스를 쉽게 교체할 수 있다는 장점을 얻게 된다.

Application 작성

1. users 테이블에 대한 VO(Value Object)객체인 UserVO 클래스(UserVo.java)를 작성한다.
 - 1-1. altibase_mybatis_spring 프로젝트의 src/main/java 디렉토리에서 마우스 오른쪽 버튼 클릭 후 New - Class를 클릭 한다.
 - 1-2. Package:에 com.altibase.test.domain을 입력하고 Name: 에 UserVo를 입력한다.



1-3. 다음의 내용을 UserVo.java 파일에 작성 한다.

```

package com.altibase.test.domain;

import java.io.Serializable;
import java.util.Date;

@SuppressWarnings("serial")
public class UserVo implements Serializable {
    private Integer userNo;
    private String userName;
    private String userContent;
    private Date regDate;

    public UserVo() {
    }

    public UserVo(Integer userNo, String userName, String userContent, Date
regDate) {
        this.userNo = userNo;
        this.userName = userName;
        this.userContent = userContent;
        this.regDate = regDate;
    }

    public Integer getUserNo() {
        return userNo;
    }

    public void setUserNo(Integer userNo) {
        this.userNo = userNo;
    }
}

```

```

public String getUsername() {
    return userName;
}

public void setUsername(String userName) {
    this.userName = userName;
}

public String getUserContent() {
    return userContent;
}

public void setUserContent(String userContent) {
    this.userContent = userContent;
}

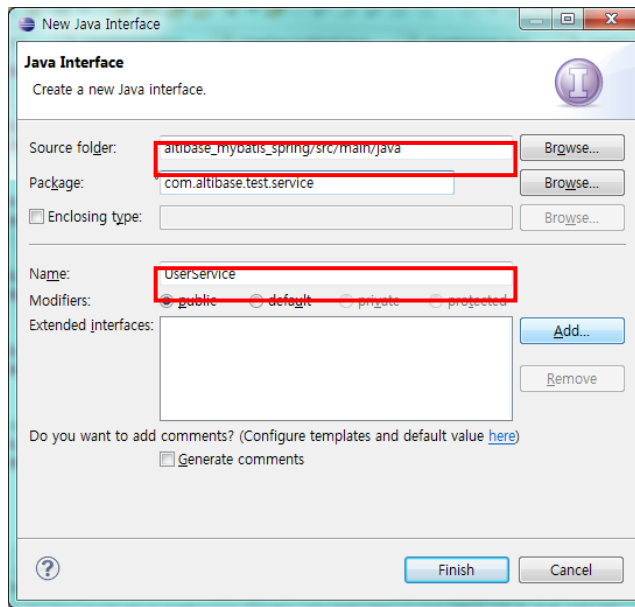
public Date getRegDate() {
    return regDate;
}

public void setRegDate(Date regDate) {
    this.regDate = regDate;
}

@Override
public String toString() {
    return "User [no=" + userNo + ", name=" + userName + ",
content=" + userContent
        + ", date=" + regDate + "];"
}
}

```

2. 실제 쿼리를 수행하는 인터페이스 및 주입 클래스 파일을 작성 한다.
 - 2-1. altibase_mybatis_spring 프로젝트의 src/main/java 디렉토리에서 마우스 오른쪽 버튼 클릭 후 New - Interface를 클릭 한다.
 - 2-2. Package:에 com.altibase.test.service를 입력하고 Name: 에 UserService를 입력한다.



2-3. 다음의 내용을 UserService.java 파일에 작성 한다.

```

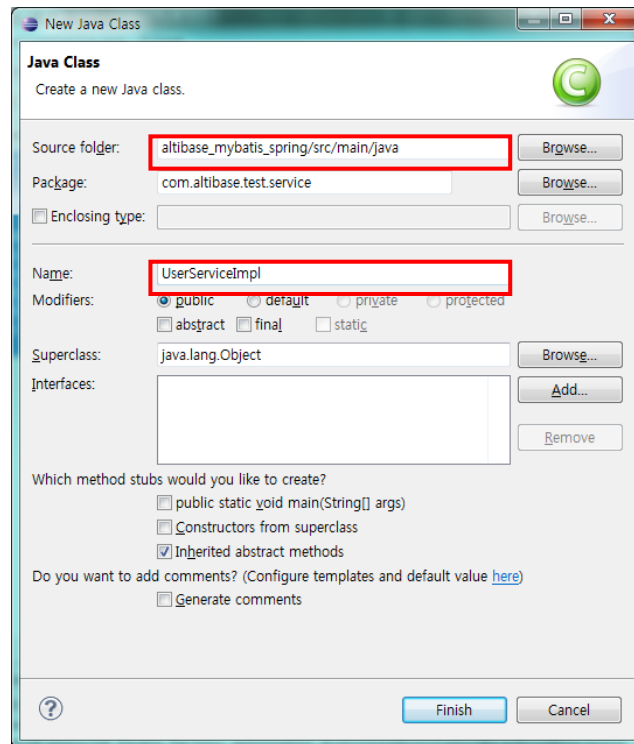
package com.altibase.test.service;

import java.util.List;
import com.altibase.test.domain.UserVo;

public interface UserService {
    public List<UserVo> getAllUser();
    public UserVo getUserByNo(int user_no);
    public Integer insertUserData(UserVo userVo);
    public Integer deleteUserByNo(int user_no);
    public Integer updateUserByNo(UserVo userVo);
}

```

2-4. 생성했던 신규 패키지(com.altibase.test.service)에 마우스 오른쪽 버튼 클릭 후 New - Class를 클릭하여 실제 로직을 처리하는 UserServiceImpl.java Class 파일을 생성 한다.



2-5. 다음의 내용을 UserMapperImpl.java 파일에 작성 한다

```

package com.altibase.test.service;

import java.util.List;

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

import com.altibase.test.domain.UserVo;
import com.altibase.test.persistence.UserMapper;

@Service("userService")
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;
    private Integer res_count;

    public List<UserVo> getAllUser() {
        List<UserVo> userVos = userMapper.getAllUser();
        return userVos;
    }

    public UserVo getUserByNo(int user_no) {
        UserVo userVo = userMapper.getUserByNo(user_no);
        return userVo;
    }

    public Integer insertUserData(UserVo userVo) {
        res_count = userMapper.insertUserData(userVo);
        return res_count;
    }

    public Integer deleteUserByNo(int user_no) {

```

```

        res_count = userMapper.deleteUserByNo(user_no);
        return res_count;
    }

    public Integer updateUserByNo(UserVo userVo) {
        res_count = userMapper.updateUserByNo(userVo);
        return res_count;
    }
}

```

3. 실제 프로그램이 수행되는 Main 클래스를 Main.java 파일에 작성 한다.

```

package com.altibase.test;

import java.util.Calendar;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.altibase.test.domain.UserVo;
import com.altibase.test.service.UserService;

public class Main {

    private static UserService userService;

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int res_count = 0;

        ApplicationContext applicationContext = new
ClassPathXmlApplicationContext("META-INF/spring/ApplicationContext.xml");
        userService =
(UserService)applicationContext.getBean("userService");

        UserVo userVo = new UserVo();

        // Insert 샘플
        userVo.setUserNo(1);
        userVo.setUserName("psy");
        userVo.setUserContent("psy_content");
        userVo.setRegDate(Calendar.getInstance().getTime());
        res_count = userService.insertUserData(userVo);
        System.out.println("insert count = " + res_count);

        // Select 샘플
        userVo = userService.getUserByNo(1);
        System.out.println("no = " + userVo.getUserNo());
        System.out.println("name = " + userVo.getUserName());
        System.out.println("content = " + userVo.getUserContent());
        System.out.println("date = " + userVo.getRegDate());

        // Update 샘플
        userVo.setUserName("updated_psy");
        userVo.setUserContent("updated_psy_content");
        res_count = userService.updateUserByNo(userVo);
        System.out.println("update count = " + res_count);
    }
}

```



```

// Delete 샘플
res_count = userService.deleteUserByNo(userVo.getUserNo());
System.out.println("delete count = " + res_count);
    }
}

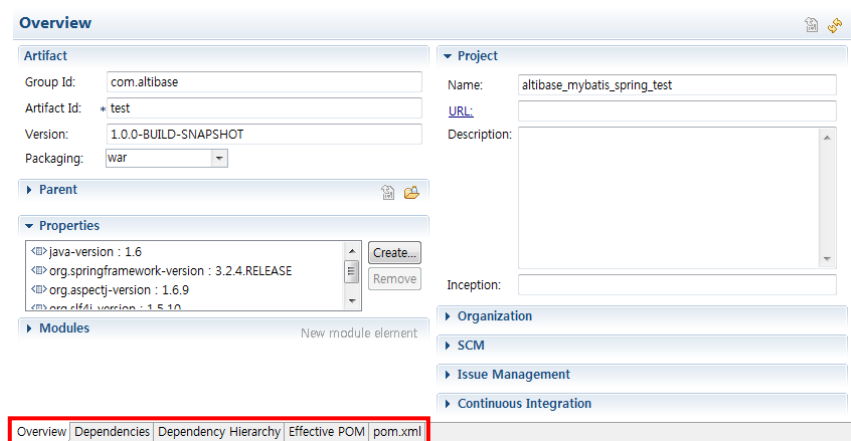
```

관련 JAR 파일 추가

Jar를 추가하는 방법은 두 가지 이다.

- Maven에 의한 라이브러리 추가
 - 기존 방식대로(properties-Java Build Path의 Add External JARS)의 라이브러리 추가
- 샘플 소스는 Maven에 의한 라이브러리 관리 방법을 선택하여 작성 되었으며, 두 번째 방법인 기존 방식은 위에서도 설명한 바 있어 본 장에선 Maven에 의한 라이브러리 관리 방법을 설명 한다.

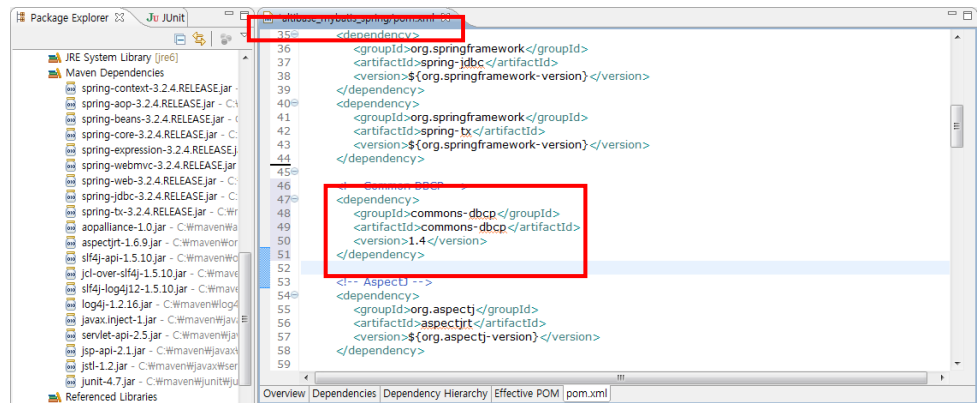
Maven을 설치 하여 Spring Template Project를 생성하면 pom.xml이라는 파일이 생성 되는데, 해당 파일이 라이브러리를 관리해 주는 파일 이다.



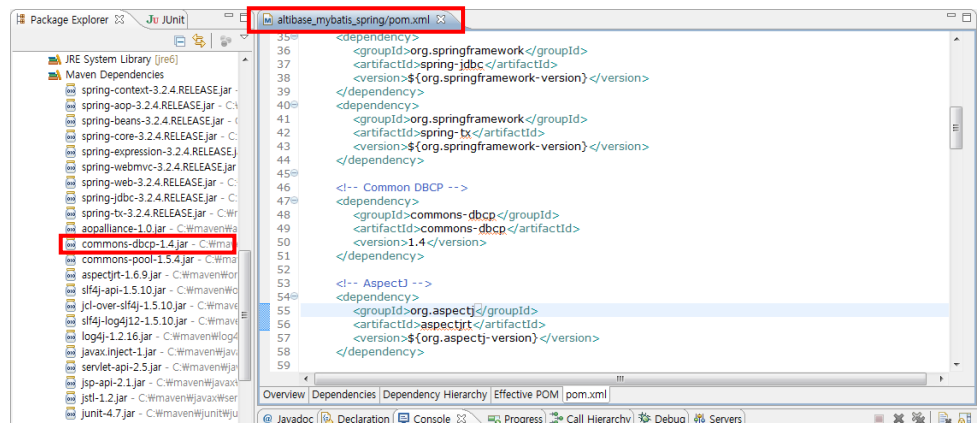
pom.xml 파일에서 library를 추가하는 방법은 두 가지이다.

- pom.xml 탭에서 태그를 직접 입력 하여 library 추가
- Dependencies 탭에서 Add 버튼으로 library 추가

태그를 직접 입력 하는 방법은, 위의 탭중 pom.xml 탭을 눌러 나타나는 xml 파일에 직접 입력하여 library를 추가 하는 방법이며 아래 화면과 같이 추가 한다. (테스트에 사용한 library는 Apache common dbcp library 이다.)



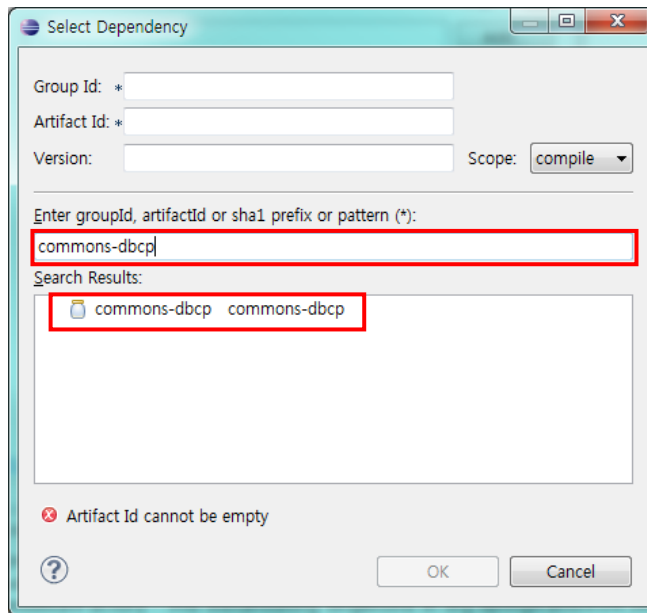
추가한 후에 xml 파일을 저장하면 Maven Dependencies에 Library가 추가된 것을 확인할 수 있다.



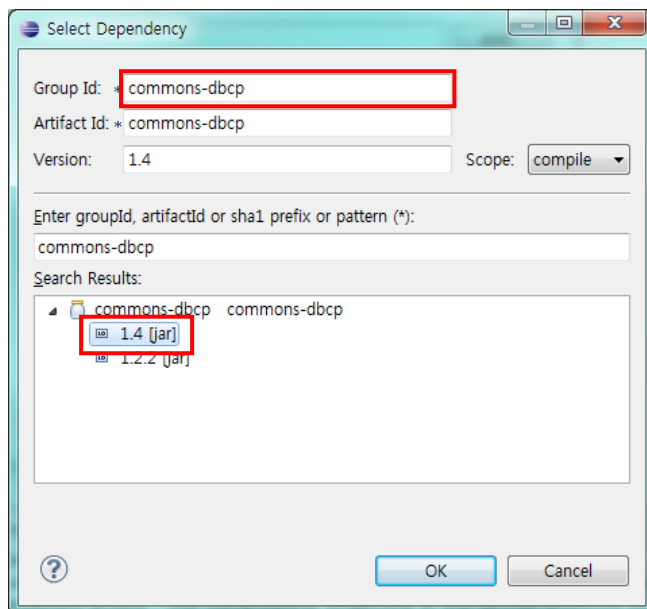
Dependencies 탭에서 Add 버튼으로 library 추가하는 방법은 다음과 같은 단계를 거쳐 추가가 가능하다.

1) pom.xml 파일에서 dependencies-Add 버튼을 클릭하면 다음의 창이 뜨게 된다.

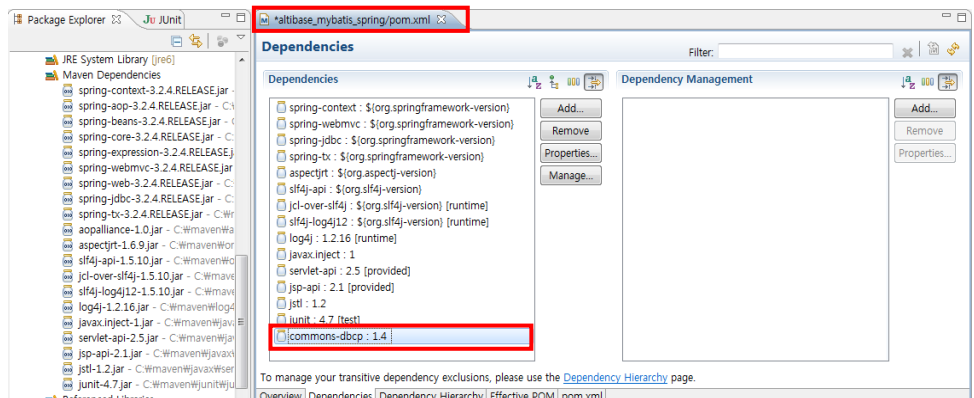
해당 창의 Enter groupId, artifactId or sha1 prefix or pattern (*): 칸에 추가하고 싶은 library의 이름을 입력하게 되면 아래와 같이 입력한 이름에 대한 라이브러리를 보여 준다.



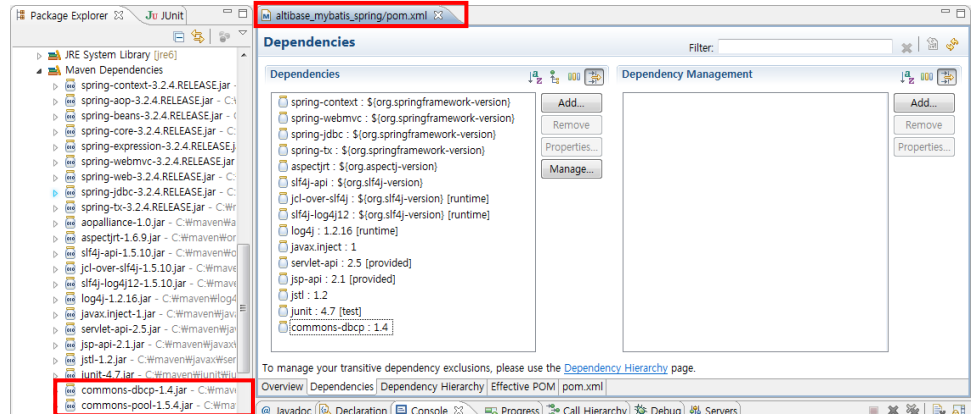
2) Search Results에 나타난 Library를 클릭하면 해당 Library에 대한 버전이 나타나게 되고 필요한 버전을 클릭한 후 OK 버튼을 클릭하면 해당 Library가 추가 된다.



라이브러리가 추가된 것을 확인할 수 있다.



파일을 저장하면 변경된 내역이 반영되면서 Library가 추가 된다.

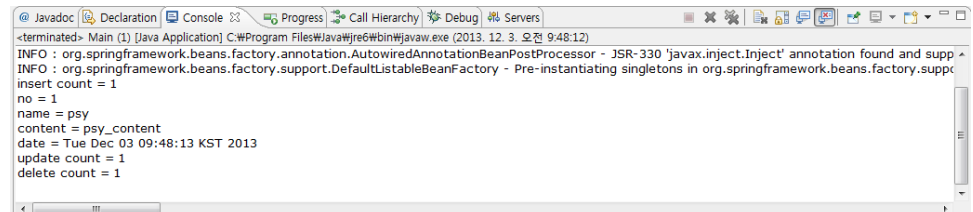


마지막에 Library가 추가된 것을 보면 commons-dbcp 말고 commons-pool도 같이 추가가 되었는데, dependencies 탭을 통해 추가하게 되면 이와 같이 관련된 Library도 같이 추가해 준다.

Application 실행

altibase_mybatis_spring 프로젝트를 실행한다.

altibase_mybatis_spring 프로젝트를 클릭한 후 메뉴에서 Run을 실행하거나 com.altibase.test 패키지의 Main.java 클래스를 더블 클릭하여 파일을 연 후에 Run을 실행 한다.



ALTIBASE[®]

알티베이스㈜

서울특별시 구로구 구로 3 동 182-13
대릉포스트 2 차 1008 호
02-2082-1000
<http://www.altibase.com>

대전사무소

대전광역시 서구 둔산동 921
주은리더스텔 901 호
042-489-0330

기술본부

서울특별시 구로구 구로 3 동 182-13
대릉포스트 2 차 908 호
02-2082-1000

기술지원센터

02-2082-1114
support@altibase.com

Altibase Support Central

<http://support.altibase.com>

Copyright © 2000~2014 ALTIBASE Corporation. All Rights Reserved.

이 문서는 정보 제공을 목적으로 제공되며, 사전에 예고 없이 변경될 수 있습니다. 이 문서는 오류가 있을 수 있으며, 상업적 또는 특정 목적에 부합하는 명시적, 묵시적인 책임이 일체 없습니다. 이 문서에 포함된 ALTIBASE 제품의 특징이나 기능의 개발, 발표 등의 시기는 ALTIBASE 재량입니다. ALTIBASE는 이 문서에 대하여 관련된 특허권, 상표권, 저작권 또는 기타 지적 재산권을 보유할 수 있습니다.