Real Alternative DBMS ALTIBASE, Since 1999

iBATIS 연동 가이드

2010. 09

Document Control

Change Record

Date	Author	Change Reference
2010-09	snkim	Created

Reviews

Date	Name (Position)

Distribution

Name	Location

목차

개요

본 문서는 iBATIS 환경, iBATIS+Spring 환경에서 ALTIBASE와 연동하는 방법에 대해 기술한다.
iBATIS 2.3.4, Spring Framework 2.5.6, ALTIBASE는 5.3.3 버전, 개발 IDE로는 Eclipse를 사용하였고, 문서 이외에 각 chapter 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_WEBSPHERE 연동가이드』
6. 『ALTIBASE_WEBLOGIC 연동가이드』
7. 『ALTIBASE_Spring 연동가이드』
8. 『ALTIBASE_HIBERNATE 연동가이드』

iBATIS 개요

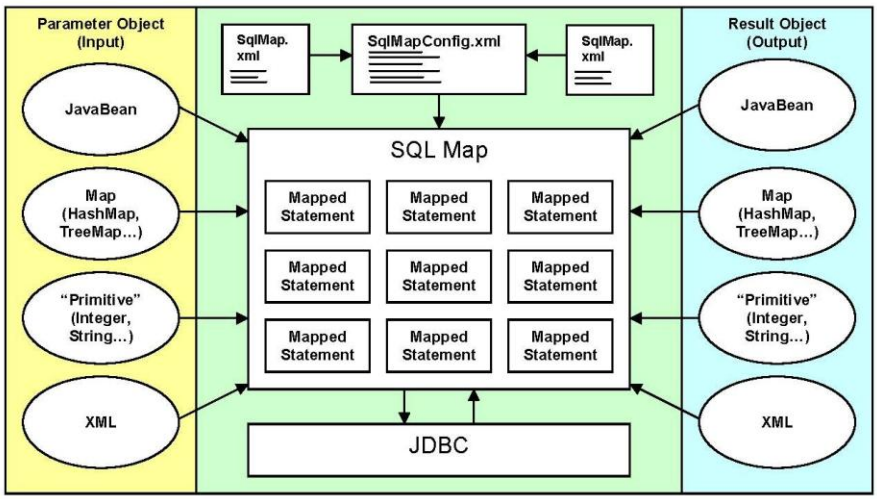
본 장에서는 iBATIS의 개념과 특징, 다운로드 및 사용 방법에 대해 살펴본다.

iBATIS란?

iBATIS란 프로그래머가 DB를 보다 편리하게 핸들링 할 수 있게 해주는 ORM(Object Relational Mapping) 프레임워크로서 DB 테이블과 JAVA 객체와의 관계를 mapping시켜 persistence logic 처리를 도와주는 역할을 한다. 즉, iBATIS를 이용하면 DB의 테이블과 JavaBean을 mapping(SqlMap XML파일)시켜 DB에 CRUD(생성, 조회, 수정, 삭제) 작업을 쉽게 할 수 있다.

기존의 JDBC를 이용하여 프로그래밍하는 방식은 프로그램 소스 안에 SQL문을 작성하였지만, iBATIS를 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 iBATIS에서 xml(SqlMap.xml)로 작성된 SQL을 DB와 어떻게 통신하는지를 간단히 보여주는 그림이다.



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

iBATIS의 보다 자세한 아키텍처는 다음의 사이트를 참고하면 된다.

http://ibatis.apache.org

iBATIS 다운로드

iBATIS를 사용하기 위해서 iBATIS 관련 jar 파일이 필요하다. 이 jar 파일은 http://ibatis.apache.org/java.cgi 사이트에서 다운로드 받을 수 있다. 다운로드 받은 파일의 압축을 풀면 압축 푼 디렉토리 안의 lib 디렉토리에 jar 파일이 존재하는데 이

jar 파일을 이용하여 iBATIS와 연동하면 된다. 만약, iBATIS 패키지가 ibatis-2.3.4.726.zip 이라면 압축을 푼 디렉토리의 lib 디렉토리에 ibatis-2.3.4.726.jar 파일이 한다.

기존 iBATIS 버전에서는 ibatis-comm.jar, ibatis-sqlmap.jar 파일들이 필요했지만, iBATIS 2.3.4 버전에서는 ibatis-comm.jar 파일과 ibatis-sqlmap.jar 파일이 ibatis-2.3.4.x.jar 파일로 통합되었다. 본 문서에서는 ibatis-2.3.4.726.jar 를 사용하였다.

iBATIS를 이용한 sample 작성

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

SqlMap 파일 작성

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

다음은 person 테이블에 CRUD를 처리하는 SqlMap XML 파일을 작성한 예제이다. (Person.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN"
"http://www.ibatis.com/dtd/sql-mapconfig-2.dtd">

<sqlMap namespace="Person">
    <resultMap id="PersonResult" class="examples.domain.Person">
            <result property="id" column="per_id" />
            <result property="name" column="per_name" />
            <result property="birthDate" column="per_birth_date" />
            <result property="weightInKilograms" column="per_weight_kg" />
            <result property="heightInMeters" column="per_height_m" />
    </resultMap>

    <select id="getPerson" parameterClass="int"
        resultClass="examples.domain.Person">
            <![CDATA[
                SELECT
                PER_ID as id,
                PER_NAME as name,
                PER_BIRTH_DATE as birthDate,
                PER_WEIGHT_KG as weightInKilograms,
                PER_HEIGHT_M as heightInMeters
                FROM PERSON
                WHERE PER_ID = #value#
            ]]>
    </select>

    <insert id="insertPerson" parameterClass="examples.domain.Person">
            <![CDATA[
                INSERT INTO
                PERSON (PER_ID, PER_NAME, PER_BIRTH_DATE,
                PER_WEIGHT_KG, PER_HEIGHT_M)
                VALUES (#id#, #name#, #birthDate#,
                        #weightInKilograms#, #heightInMeters#)
            ]]>
    </insert>

    <update id="updatePerson" parameterClass="examples.domain.Person">
            <![CDATA[
                UPDATE PERSON
```

```
                        SET PER_NAME = #name#,
                        PER_BIRTH_DATE = #birthDate#,
                        PER_WEIGHT_KG = #weightInKilograms#,
                        PER_HEIGHT_M = #heightInMeters#
                        WHERE PER_ID = #id#
                ]]>
    </update>

    <delete id="deletePerson" parameterClass="int">
                <![CDATA[
                        DELETE PERSON
                        WHERE PER_ID = #id#
                ]]>
    </delete>

    <select id="getAllPersons"    resultMap="PersonResult">
                <![CDATA[
                        SELECT * FROM person
                ]]>
    </select>
</sqlMap>
```

<resultMap> 태그에는 SELECT문을 실행 후 ResultSet에 담길 데이터들의 Map 객체에 대해 정의하고, <insert>, <update>, <delete>, <select> 태그에는 CRUD의 작업에 대한 각각의 SQL문을 정의한다.

각각의 태그에 대한 보다 자세한 설명은 http://ibatis.apache.org 사이트를 참고하거나 첨부된 문서 iBATIS-SqlMaps-2-ko.pdf 파일을 참고하면 된다.

SqlMapConfig 파일 작성

SqlMapConfig 파일은 DB 연결을 위한 dataSource, SqlMap 파일의 경로, 그 외 SqlMapClient를 제어할 property들을 작성하는 SQL Maps 설정파일이다.

다음은 SqlMapConfig 파일(SqlMapConfigExample.xml) 예제이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config
2.0//EN"
"http://www.ibatis.com/dtd/sql-map-config-2.dtd">

<sqlMapConfig>

    <properties resource="db.properties" />

    <settings
        cacheModelsEnabled="true"
        enhancementEnabled="true"
        lazyLoadingEnabled="true"
        maxRequests="32"
        maxSessions="10"
        maxTransactions="5"
        useStatementNamespaces="false"
    />

    <transactionManager type="JDBC" >
          <dataSource type="SIMPLE">
              <property name="JDBC.Driver" value="${driver}"/>
              <property name="JDBC.ConnectionURL" value="${url}"/>
```

```
            <property name="JDBC.Username" value="${username}"/>
            <property name="JDBC.Password" value="${password}"/>
            <property name="Pool.MaximumActiveConnections" value="10"/>
            <property name="Pool.MaximumIdleConnections" value="5"/>
            <property name="Pool.MaximumCheckoutTime" value="120000"/>
            <property name="Pool.TimeToWait" value="500"/>
            <property name="Pool.PingQuery" value="select 1 from dual"/>
            <property name="Pool.PingEnabled" value="false"/>
            <property name="Pool.PingConnectionsOlderThan" value="1"/>
            <property name="Pool.PingConnectionsNotUsedFor" value="1"/>
        </dataSource>
    </transactionManager>

    <sqlMap resource="Person.xml" />

</sqlMapConfig>
```

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

각각의 태그에 대한 보다 자세한 설명은 http://ibatis.apache.org 사이트를 참고하거나 첨부된 문서 iBATIS-SqlMaps-2-ko.pdf 파일을 참고하면 된다.

SqlMapConfig 파일 작성 – iBatis.Net 연동 시

ODBC를 통한 접속 시 SqlMap.config 의 알티베이스 연결 설정 방법을 간단히 설명한다.

먼저 알티베이스 ODBC Driver 설치 및 ODBC 데이터 원본 관리자에서 사용자 DSN 을 추가해야 한다.

ODBC 설치 및 설정 방법은 기술문서 'ALTIBASE_Windows_ODBC_개발가이드' 를 참고한다.

```
<!-- Database connection information -->
<database>
    <provider name="Odbc2.0"/>
    <dataSource name="Altibase" connectionString="DSN=Altibase5;USER
ID=sys;PASSWORD=manager"/>
</database>
```

providers.config에 정의된 여러 DBMS Provider 중에서 알티베이스 접속 시 사용할 Provider 는 Odbc2.0 이다. <provider>태그에 Odbc2.0 을 작성한다.

<dataSource> 태그에서 connectionString에 ODBC 데이터 원본 관리자에서 추가한 DSN을 입력한다.

Application 작성

Application에서 SqlMapClient 인스턴스를 이용하여 DB 테이블에 Mapping된 객체와 연동하여 CRUD 작업을 처리할 수 있다.

iBATIS를 이용하여 DB와 연동하기 위해서는 먼저 SqlMapConfig 파일을 통해 SqlMapClient 객체를 얻어와야 한다. 이후 SqlMapClient 객체를 통해 CRUD에 해당하는 메소드를 호출하여 DB와 통신하면 된다.

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

예) SimpleConnection의 PersonApp.java

```
…
String resource ="SqlMapConfigExample.xml";
Reader reader = Resources.getResourceAsReader(resource);
SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);

//insert Person
Person newPerson1 = new Person();
…
sqlMap.insert ("insertPerson", newPerson1);
…

//select all Persons
List<Person> list = (List<Person>)sqlMap.queryForList("getAllPersons");
…

//update Person
sqlMap.update("updatePerson", newPerson1);
…

//get Person
Person person = (Person) sqlMap.queryForObject ("getPerson", personPk);
…

//delete Person
sqlMap.delete ("deletePerson", new Integer(1));
…
```

먼저 SqlMapConfig 파일을 읽어 들여 SqlMapClient 객체를 얻어온다. (

```
String resource ="SqlMapConfigExample.xml";
Reader reader = Resources.getResourceAsReader(resource);
SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
```

)

이후 CRUD에 해당하는 SqlMapClient 클래스의 각각의 메소드를 호출한다. (

sqlMap.insert(), sqlMap.queryForList(), sqlMap.queryForObject(),sqlMap.update(), sqlMap.delete()

)

각 메소드의 자세한 설명은 http://ibatis.apache.org 사이트를 참고하거나 첨부된 문서 iBATIS-SqlMaps-2-ko.pdf 파일을 참고하면 된다.

ALTIBASE 연동

iBATIS에서 ALTIBASE를 연동하는 위해서는 ALTIBASE JDBC Driver를 setting하고 SqlMapConfig 파일에 ALTIBASE를 위한 dataSource를 지정하면 된다. 본 장에서는 ALTIBASE JDBC Driver를 얻는 방법, JDBC Driver를 설정하는 방법, SqlMapConfig에 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 = 5.3.3.13 for JavaVM v1.4, CMP:5.6.1, $Revision: 14502 $ Jan 13 2010 14:35:28
```

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

```
$ altibase -v
version 5.3.3.13 XEON_LINUX_redhat_Enterprise_AS4-64bit-5.3.3.13-release-GCC3.4.6 (xeon-redhat-linux-gnu) Jan 13 2010 14:35:30, binary db version 5.4.1, meta version 5.6.1, cm protocol version 5.6.1, replication protocol version 5.4.1
```

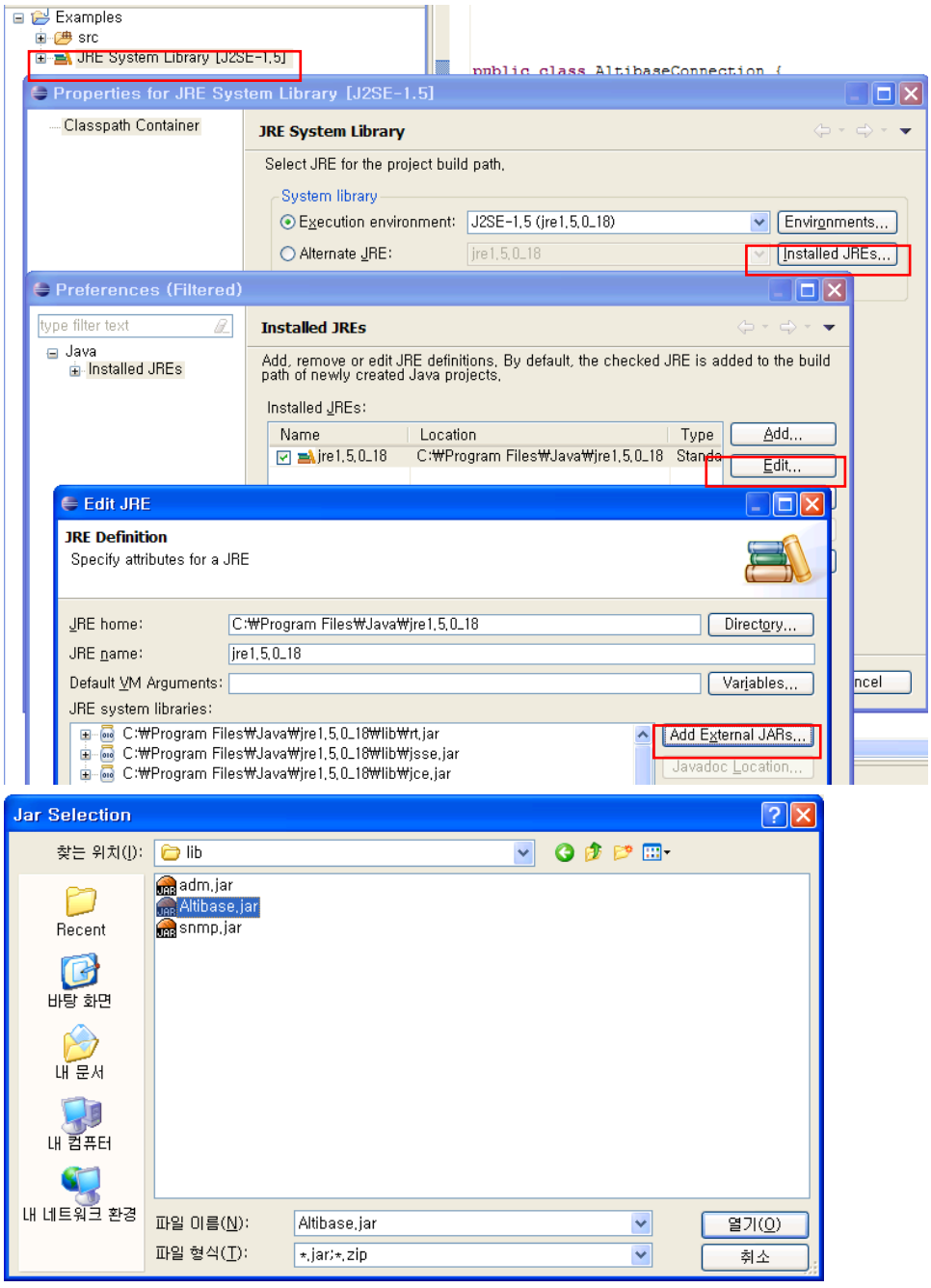
버전이 UP 되면서 JDBC 관련 버그가 fix되었을 가능성이 있으므로, 일반적으로 ALTIBASE DB Server의 버전과 같거나 이 보다 더 최신의 ALTIBASE JDBC Driver 파일을 사용하는 것을 권장한다.

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 – JRE System Library [J2SE-1.5] - Properties – Installed JREs – 항목 중 jre를 클릭 – Edit – Add External JARs 를 클릭하여 ALTIBASE JDBC Driver인 Altibase.jar를 추가한다.



SqlMapConfig 파일에 dataSource를 설정하여 ALTIBASE와 연동

SqlMapConfig 파일의 <transactionManager> 태그에 ALTIBASE 용 property를 지정하여 ALTIBASE와 연결하면 된다. 이 때 SqlMapConfig 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 SqlMapConfig 파일에서 사용하는 예제이다.

예) SimpleConnection의 db.properties 파일

```
driver=Altibase.jdbc.driver.AltibaseDriver
url=jdbc:Altibase://192.168.1.35:21129/mydb
username=sys
password=manager
```

이 파일에 설정된 각각의 값의 의미는 다음과 같다.

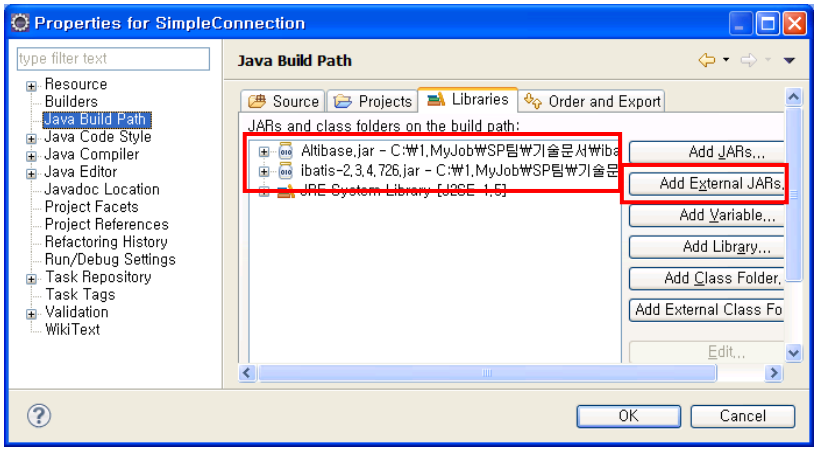
Property	설명
driver	ALTIBASE JDBC driver class Name
url	ALTIBASE와 연결을 위한 Connection string정보 jdbc:Altibase://IP:port_no/db_name" 형태로 기입
username	데이터베이스 계정
password	데이터베이스 패스워드

예) SimpleConnection의 SqlMapConfigExample.xml 파일

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

  <transactionManager type="JDBC" >
    <dataSource type="SIMPLE"> -- SIMPLE은 내장된 트랜잭션 관리자이름
      <property name="JDBC.Driver" value="${driver}"/>
      <property name="JDBC.ConnectionURL" value="${url}"/>
      <property name="JDBC.Username" value="${username}"/>
      <property name="JDBC.Password" value="${password}"/>
    </dataSource>
  </transactionManager>

  <sqlMap resource="Person.xml" />

</sqlMapConfig>
```

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, ibatis-2.3.4.x.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)&amp;
  ConnectionRetryCount=1&amp;ConnectionRetryDelay=1&amp;
  SessionFailOver=on&amp;LoadBalance=off
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
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 사용자에 의해 재처리되어야 한다)

위의 예제 FailOverSample 프로젝트를 실행하기 위해서는 “SqlMapConfig 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 5 와 ALTIBASE 4 혹은 ALTIBASE 3 와 동시에 연결할 수 있도록 ALTIBASE 5 버전 전용의 JDBC Driver(Altibase5.jar)를 제공한다. 이 Driver를 이용하면 ALTIBASE 5 – ALTIBASE 4, 혹은 ALTIBASE 5 – ALTIBASE 3, 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.

iBATIS에 다른 버전의 ALTIBASE 와 연동하기 위해서는 각 버전에 해당하는 SqlMapConfig 파일을 별도로 작성하여 어플리케이션에서 각각의 SqlMapConfig 파일을 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) MultiVersionConnection의 db.properties1 파일

ALTIBASE 5 버전에 대한 설정

```
driver=Altibase5.jdbc.driver.AltibaseDriver
url=jdbc:Altibase://192.168.6.224:21129/mydb
username=sys
password=manager
```

예) MultiVersionConnection의 db.properties2 파일

ALTIBASE 5 이전 버전에 대한 설정

```
driver=Altibase.jdbc.driver.AltibaseDriver
url=jdbc:Altibase://192.168.1.35:21129/mydb
username=sys
password=manager
```

예) MultiVersionConnection의 SqlMapConfigExample1.xml파일

ALTIBASE 5 버전에 대한 설정

```
<sqlMapConfig>
    <properties resource="db.properties1" />
    <transactionManager type="JDBC" >
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="${driver}"/>
            <property name="JDBC.ConnectionURL" value="${url}"/>
            <property name="JDBC.Username" value="${username}"/>
            <property name="JDBC.Password" value="${password}"/>
        </dataSource>
    </transactionManager>
    <sqlMap resource="Person.xml" />
</sqlMapConfig>
```

예) MultiVersionConnection의 SqlMapConfigExample2.xml파일

ALTIBASE 5 이전 버전에 대한 설정

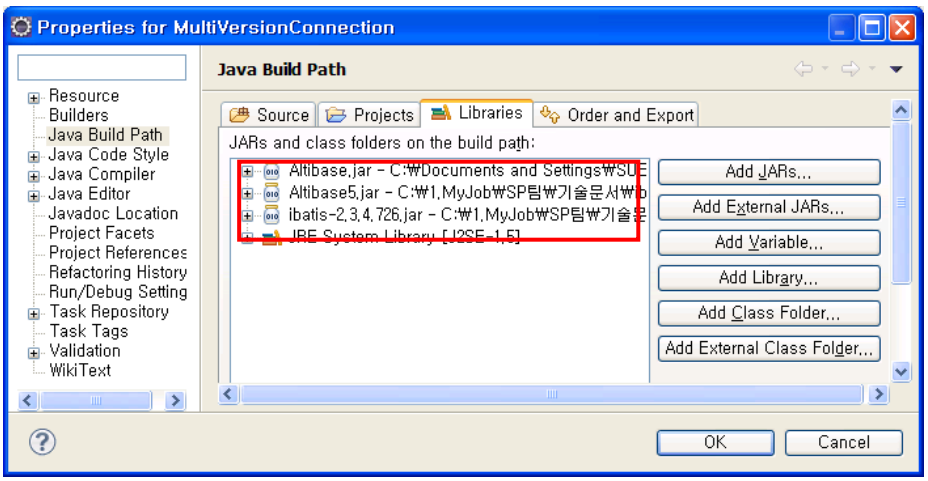
```
<sqlMapConfig>
    <properties resource="db.properties2" />
    <transactionManager type="JDBC" >
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="${driver}"/>
            <property name="JDBC.ConnectionURL" value="${url}"/>
            <property name="JDBC.Username" value="${username}"/>
            <property name="JDBC.Password" value="${password}"/>
        </dataSource>
    </transactionManager>
    <sqlMap resource="Person.xml" />
</sqlMapConfig>
```

예) MultiVersionConnection의 PersonApp.java파일

```
…
String resource1 ="SqlMapConfigExample1.xml";
Reader reader1 = Resources.getResourceAsReader(resource1);
SqlMapClient sqlMap1 = SqlMapClientBuilder.buildSqlMapClient(reader1);

String resource2 ="SqlMapConfigExample2.xml";
Reader reader2 = Resources.getResourceAsReader(resource2);
SqlMapClient sqlMap2 = SqlMapClientBuilder.buildSqlMapClient(reader2);
…
```

위의 PersonApp.java 예제를 보면 Altibase5.jdbc.driver.AltibaseDriver를 Altibase.jdbc.driver.AltibaseDriver보다 먼저 로딩하기 위해 Altibase5.jdbc.driver.AltibaseDriver를 JDBC.Driver로 사용하는 SqlMapConfigExample1.xml 파일을 먼저 읽어드리고 있다. 반드시 ALTIBASE5 전용의 드라이버를 먼저 로딩해야 한다.

예제에 포함된 MultiVersionConnection 프로젝트를 실행하기 위해서는 기존에 사용했던 ibatis-2.3.4.x.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.



Procedure/Function 호출

iBATIS에서 DB에 생성한 Stored Procedure/Function을 호출할 경우에는 SqlMap 파일에 Stored Procedure/Function에 정의된 parameter에 대한 정보를 설정해주고, <procedure> 태그에 CallableStatement에 적용되는 Procedure/Function을 호출하는 문장을 정의해주면 된다.

다음은 Stored Procedure/Function을 호출하는 예제이다.

예) ProcedureSample의 Procedure/Function 생성 구문

```
CREATE OR REPLACE PROCEDURE sum_proc
(
        p_num1 IN NUMBER,
        p_num2 IN NUMBER,
        p_num3 OUT NUMBER
)
AS
BEGIN
        p_num3 := p_num1 + p_num2;
END;
/

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) Procedure의 Procedure.xml(SqlMap) 파일

```
<sqlMap namespace="Procedure">

  <parameterMap id="ProcedureParam" class="java.util.Map">
      <parameter property="p_num1" jdbcType="NUMERIC"
        javaType="int" mode="IN"    />
      <parameter property="p_num2" jdbcType="NUMERIC"
        javaType="int" mode="IN"    />
      <parameter property="p_num3" jdbcType="NUMERIC"
        javaType="int" mode="OUT"/>
 </parameterMap>


 <parameterMap id="FunctionParam" class="java.util.Map">
      <parameter property="p_num3" jdbcType="NUMERIC"
        javaType="int" mode="OUT"/>
      <parameter property="p_num1" jdbcType="NUMERIC"
        javaType="int" mode="IN" />
      <parameter property="p_num2" jdbcType="NUMERIC"
        javaType="int" mode="IN" />
 </parameterMap>
```

```
<procedure id="sumProc" parameterMap="ProcedureParam" >
        {call sum_proc(?,?,?)}
</procedure>

<procedure id="sumFunc" parameterMap="FunctionParam"    >
        {call ? := sum_func(?,?)}
 </procedure>

</sqlMap>
```

<paramaterMap> 태그에 Procedure/Function의 파라미터에 대한 타입과 IN/OUT 설정을 정의하고 <procedure> 태그에서 parameterMap 속성에 <paramaterMap> 태그의 id 값을 명시해준다. 그리고 <procedure> 태그에 Procedure/Function을 호출하는 문장을 작성해준다.

ProcedureSample 예제를 실행하기 위해서는 "SqlMapConfig 파일에 dataSource를 설정하여 ALTIBASE와 연동"과 마찬가지로 Altibase.jar, ibatis-2.3.4.x.jar 파일이 필요하다.

iBATIS, Spring, ALTIBASE 연동

ALTIBASE와 연동하기 위해서는 iBATIS에 dataSource를 지정할 수도 있고, Spring에 dataSource를 지정할 수도 있다. 본 장에서는 이 두 방법을 이용하여 ATLIBASE와 연동하는 방법에 대해 설명한다.

Spring에 dataSource를 설정하는 경우

iBATIS와 Spring을 함께 사용하기 위해서는 Spring의 applicationContext.xml 파일에 iBATIS의 SqlMapClientFactoryBean bean을 지정해 주면 된다. 이 후 각각의 DAO bean에 SqlMapClientFactoryBean bean을 참조하도록 설정해주면 다른 bean 에서 SqlMapClient 객체를 사용하여 CRUD에 해당하는 메소드들을 호출할 수 있다.

iBATIS, Spring을 연동한 환경에서 Spring에 dataSource를 설정하는 방법은『ALTIBASE_Spring_연동가이드』문서에서 설명한 방법들 중 하나를 선택하여 applicationContext.xml에 dataSource를 지정해주고, iBATIS와 연동을 위해 iBATIS의 SqlMapClientFactoryBean bean을 지정해주면 된다. 이 때 SqlMapClientFactoryBean bean의 configLocation property에 SqlMapConfig 파일의 이름을 명시해준다.

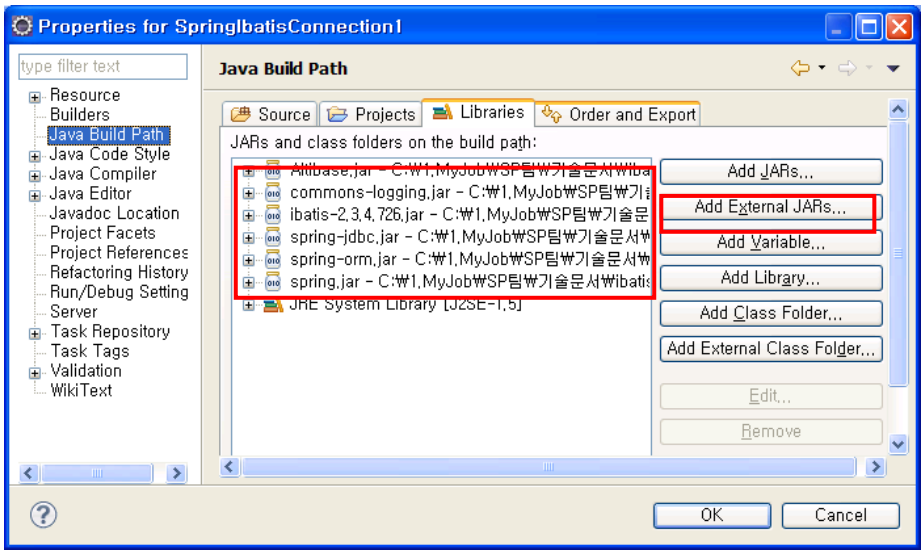
다음은 applicationContext.xml 파일에서 dataSource와 SqlMapClientFactoryBean bean을 지정해주는 예제이다.

예) SpringIbatisConnection1 의 applicationContext.xml 파일

```
…
<!-- DriverManagerDataSource 클래스를 이용한 데이터 소스 설정 -->
<bean id="dataSource"
 class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <!--  JDBC Driver 클래스명 설정 -->
    <property name="driverClassName" value="Altibase.jdbc.driver.AltibaseDriver"/>
    <!-- connection url-->
    <property name="url" value="jdbc:Altibase://192.168.1.35:21129/mydb"/>
    <!-- DB 사용자 계정 설정 -->
    <property name="username" value="sys"/>
    <!-- DB 사용자 패스워드 설정 -->
    <property name="password" value="manager" />
</bean>

<bean id="sqlMapClient"
 class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="SqlMapConfigExample.xml"/>
</bean>

<!-- DAO 클래스의 bean 설정 -->
<bean id="personDao" class="examples.domain.PersonDao">
    <property name="sqlMapClient" ref="sqlMapClient"/>
</bean>
…
```

위의 예제 SpringIbatisConnection1 프로젝트를 실행하기 위해서는 Altibase.jar, ibatis-2.3.4.x.jar 파일과 spring-jdbc.jar, spring-orm.jar, spring.jar, commons-logging.jar 파일이 필요하다. spring-jdbc.jar, spring-orm.jar, spring.jar, commons-logging.jar 파일은 Spring

Framework에 포함된 파일이다. 자세한 디렉토리 위치는
『ALTIBASE_Spring_연동가이드』 문서를 참고하면 된다.



iBATIS에 dataSource를 설정하는 경우

iBATIS, Spring을 연동한 환경에서 iBATIS에 dataSource를 설정하는 방법은 위의 "SqlMapConfig 파일에 dataSource를 설정하여 ALTIBASE와 연동" 부분에서 설명한 방법과 동일한 방법으로 SqlMapConfig 파일에 <transactionManager>에 ALTIBASE 연결을 위한 property를 지정하면 된다.

다만, Spring에서 iBATIS를 연동하기 위해 위의 "Spring에서 dataSource를 설정하는 경우"와 마찬가지로 iBATIS의 SqlMapClientFactoryBean bean 을 applicationContext.xml 파일에 지정해줘야 한다.

다음은 iBATIS와 Spring 연동 환경에서 iBATIS에 dataSource를 설정하여 ALTIBASE와 연동하는 예제이다.

예) SpringIbatisConnection2 의 applicationContext.xml 파일

```
…
<bean id="sqlMapClient"
 class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
    <property name="configLocation" value="SqlMapConfigExample.xml"/>
</bean>

<!-- DAO 클래스의 bean 설정 -->
<bean id="personDao" class="examples.domain.PersonDao">
    <property name="sqlMapClient" ref="sqlMapClient"/>
</bean>
 …
```

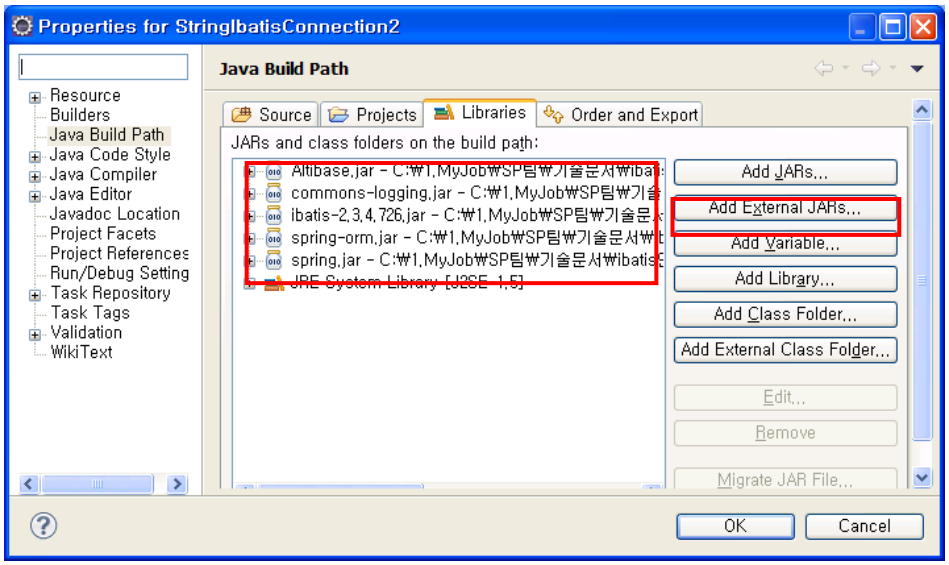
예) SpringIbatisConnection2 의 SqlMapConfigExample.xml 파일

```
<sqlMapConfig>
    <properties resource="db.properties" />
    <transactionManager type="JDBC" >
        <dataSource type="SIMPLE">
```

```
            <property name="JDBC.Driver" value="${driver}"/>
            <property name="JDBC.ConnectionURL" value="${url}"/>
            <property name="JDBC.Username" value="${username}"/>
            <property name="JDBC.Password" value="${password}"/
        </dataSource>
    </transactionManager>

    <sqlMap resource="Person.xml" />

</sqlMapConfig>
```

위의 예제를 보면 dataSource를 SqlMapConfigExample.xml에 설정하고 applicationContext.xml 파일에는 이 SqlMapConfigExample.xml 파일을 읽어 SqlMapClientFactoryBean bean을 설정하고 있는 것을 확인할 수 있다.

SpringIbatisConnection2 예제를 실행하기 위해서는 Altibase.jar, ibatis-2.3.4.x.jar 파일과 spring-orm.jar, spring.jar, commons-logging.jar 파일이 필요하다. spring-orm.jar, spring.jar, commons-logging.jar 파일은 Spring Framework에 포함된 파일이다. 자세한 디렉토리 위치는 『ALTIBASE_Spring_연동가이드』 문서를 참고하면 된다.



ALTIBASE의 ConnectionPool을 이용

ALTIBASE에서 제공하는 AltibaseConnectionPoolDataSource 클래스를 사용하면 ALTIBASE의 ConnectionPool을 이용할 수 있는데, iBATIS, Spring 연동 환경에서는 Spring의 applicationContext.xml 파일에 AltibaseConnectionPoolDataSource 클래스를 이용하여 dataSource bean을 정의하면 된다. Spring과 연동 없이 iBATIS에서만 ALTIBASE의 ConnectionPool을 사용할 수 있는 방법은 없다는 것을 주의해야 한다. 자세한 내용은 『ALTIBASE_Spring_연동가이드』 문서를 참고하여 설정하면 된다.

예) 『ALTIBASE_Spring_연동가이드』에 첨부된 AltibaseConnectionPool의 applicationContext.xml 파일

```
…
<bean id="dataSource"
        class="Altibase.jdbc.driver.AltibaseConnectionPoolDataSource">
  <!-- connection url-->
   <property name="url" value="jdbc:Altibase://192.168.1.35:21129/mydb"/>
```

```
   <!-- DB 사용자 계정 설정 -->
   <property name="user" value="sys"/>
   <!-- DB 사용자 패스워드 설정 -->
   <property name="password" value="manager" />
</bean>
…
```

트랜잭션 관리

iBATIS에서 DB와 연동할 경우 SqlMapConfig 파일의 <transactionManager> 에 dataSource를 지정하면 SqlMap 파일에서 정의한 각각의 CRUD 메소드가 호출될 때 자동으로 setAutoCommit(false); 가 호출이 된다. 이후 해당 메소드가 종료되면 트랜잭션은 commit이 되고 다시 default autocommit 모드로 바뀌게 된다.

또한 프로그래머가 어플리케이션에서 명시적으로 트랜잭션을 관리할 수 도 있다.

만약, iBATIS와 Spring을 연동하였다면 Spring에서도 트랜잭션을 관리할 수 있다.

본 장에서는 이러한 트랜잭션 관리 방법들을 소개한다.

iBATIS에서 트랜잭션 관리

iBATIS에서 DB와 연동할 경우 SqlMapConfig 파일의 <transactionManager> 에 dataSource를 지정하면 SqlMap 파일에서 정의한 각각의 CRUD 메소드가 호출될 때 자동으로 setAutoCommit(false); 가 호출이 된다. 이후 해당 메소드가 종료되면 트랜잭션은 commit이 되고 다시 default autocommit 모드로 바뀌게 된다.

또한 어플리케이션에서 프로그래머가 직접 트랜잭션을 관리할 수 있다.

이때, 트랜잭션을 시작하기 위해서는 SqlMapClient의 startTransaction() 메소드를 호출하고, commit 하고자 할 경우에는 commitTransaction () 메소드, commit 없이 rollback 하고자 할 경우에는 endTransaction() 메소드를 호출하면 된다.

다음은 어플리케이션에서 명시적으로 트랜잭션을 처리하는 예제이다..

예) TransactionSample의 PersonApp.java 파일

```
…
SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);

//start transaction
sqlMap.startTransaction();

//insert Person
Person newPerson1 = new Person();
…
sqlMap.insert ("insertPerson", newPerson1);

//commit
sqlMap.commitTransaction ();

Person newPerson2 = new Person();
…
sqlMap.insert ("insertPerson", newPerson2);

// selected 2 rows.
List<Person> list = (List<Person>)sqlMap.queryForList("getAllPersons");
System.out.println("Selected "+list.size()+" records.");
for(int i=0; i< list.size();i++){
    System.out.println(list.get(i));
}

//rollback
sqlMap.endTransaction();

// selected only 1 row.
```

```
list = (List<Person>)sqlMap.queryForList("getAllPersons");
System.out.println("Selected "+list.size()+" records.");
for(int i=0; i< list.size();i++){
    System.out.println(list.get(i));
}
…
```

위의 예제 TransactionSample 프로젝트를 실행하기 위해서는 “SqlMapConfig 파일에 dataSource를 설정하여 ALTIBASE와 연동”과 마찬가지로 Altibase.jar, ibatis-2.3.4.x.jar 파일이 필요하다.

Spring에서 트랜잭션 관리

iBATIS, Spring 을 연동한 환경이라면, Spring에서 트랜잭션을 관리할 수 있다. 본 문서에서는 Spring의 applicationContext.xml 파일에 선언적으로 트랜잭션을 관리/처리하는 예제(LobSpringIbatisSample 예제)를 포함하고 있다. 보다 자세한 내용은『ALTIBASE_Spring_연동가이드』문서를 참고하면 된다.

iBATIS 연동 시 고려사항

Spring에서 ALTIBASE에 연동할 경우 고려해야 할 사항에 대해 설명한다.

LOB 데이터 처리

iBATIS에서 LOB을 처리하기 위해서는 SqlMap 파일에 parameter와 result에 대한 정보를 setting하는 부분에 반드시 jdbcType을 CLOB/BLOB이라고 명시를 해줘야 한다. 그렇지 않을 경우에 길이제한에 의해 올바르지 않은 데이터를 입력하거나 잘못된 데이터를 질의할 수 있다. 혹은 Invalid length등의 에러가 발생할 수도 있다. 따라서 반드시 parameter와 result에 대한 mapping을 CLOB/BLOB으로 지정해줘야 한다.

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

예) LobSample의 LobSample.xml(SqlMap) 파일

```
<sqlMap namespace="LobSample">
    <resultMap id="LobSampleResult" class="com.altibase.lob.LobSample">
      <result property="lob_id" column="lob_id" />
      <result property="lobcolumn" column="lobcolumn"
              jdbcType="CLOB" javaType="java.lang.String" />
    </resultMap>

    <parameterMap id="LobSampleParam" class="com.altibase.lob.LobSample">
       <parameter property="lob_id" />
       <parameter property="lobcolumn"
                  jdbcType="CLOB" javaType="java.lang.String" />
    </parameterMap>

    <select id="getLobSample" parameterClass="int" resultMap="LobSampleResult">
                  SELECT lob_id, lobcolumn
                  FROM lobsample
                  WHERE lob_id = #value#
    </select>

    <insert id="insertLobSample" parameterMap="LobSampleParam">
                  INSERT INTO     lobsample (lob_id,lobcolumn)
                  VALUES (?,?)
    </insert>

          …
</sqlMap>
```

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 바꿔주기 때문에 LOB 처리 시 따로 고려할 사항은 없다. 하지만, iBATIS와 Spring을 함께 연동할 경우 Spring에서 트랜잭션을 관리해준다면 LOB을 처리하기 위해서는 반드시 TransactionManager bean을 명시해줘야 한다.

또, Spring에서 선언적으로 트랜잭션을 처리하는 경우에는 propagation을 PROPAGATION_REQUIRED, PROPAGATION_REQUIRES_NEW, PROPAGATION_NESTED 중 하나로 지정해줘야 한다.

만약 TransactionManager를 지정해주지 않았거나, 또는 선언적 트랜잭션을 사용하는데 propagation을 위에 설명한 값 이외의 다른 값으로 지정했을 경우에는 LOB 데이터 조회 시 null 값이 리턴 되거나, "java.sql.SQLException: [0]:LobLocator can not span the transaction 101858625." 과 같은 에러가 발생한다.

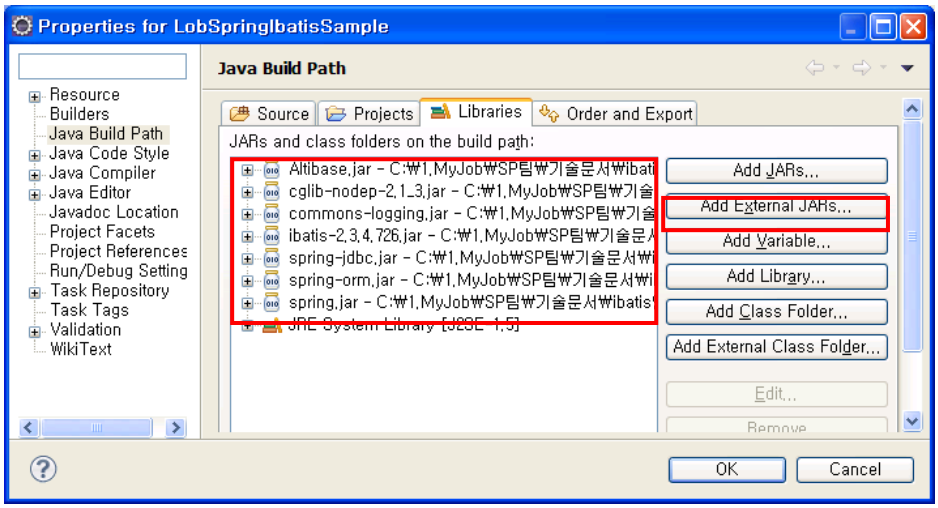
그리고 LOB 데이터를 입력 시에도 "java.sql.SQLException: [0]:LobLocator can not span the transaction 101858625." 에러가 발생하게 된다.

다음의 예제는 Spring의 applicationContext.xml에서 선언적으로 트랜잭션을 처리하여 LOB 데이터를 처리하는 예제이다. 선언적으로 트랜잭션을 처리하는 방법에 대한 자세한 설명은 『ALTIBASE_Spring_연동가이드』문서를 참고하면 된다.

예) LobSpringIbatisSample의 applicationContext.xml 파일

```
…
<bean id="dataSource"
   class="org.springframework.jdbc.datasource.DriverManagerDataSource">
   <property name="driverClassName" value="Altibase.jdbc.driver.AltibaseDriver"/>
   <property name="url" value="jdbc:Altibase://192.168.1.35:21129/mydb"/>
   <property name="username" value="sys"/>
   <property name="password" value="manager" />
</bean>

<bean id="sqlMapClient"
    class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
    <property name="dataSource" ref="dataSource"/>
    <property name="configLocation" value="SqlMapConfigExample.xml"/>
</bean>
<bean id="transactionManager"
    class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>

<bean id="txProxyTemplate" abstract="true"
   class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
   <property name="transactionManager"   ref="transactionManager" />
   <property name="transactionAttributes">
       <props>
           <prop key="insert*">PROPAGATION_REQUIRED</prop>
           <prop key="update*">PROPAGATION_REQUIRED</prop>
           <prop key="delete*">PROPAGATION_REQUIRED</prop>
           <prop key="get*">PROPAGATION_REQUIRED</prop>
     </props>
   </property>
</bean>

<bean id="lobSampleDao" parent="txProxyTemplate">
   <property name="target">
       <bean class="com.altibase.lob.LobSampleDao">
           <property name="sqlMapClient" ref="sqlMapClient"/>
       </bean>
   </property>
</bean>
…
```

위의 LobSpringIbatisSample 프로젝트를 실행하기 위해서는 Altibase.jar, ibatis-2.3.4.x.jar 파일과 spring-jdbc.jar, spring-orm.jar, spring.jar, commons-logging.jar, cglib-nodep-x.x.jar 파일이 필요하다.



부록

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

DB 테이블 및 시퀀스 생성

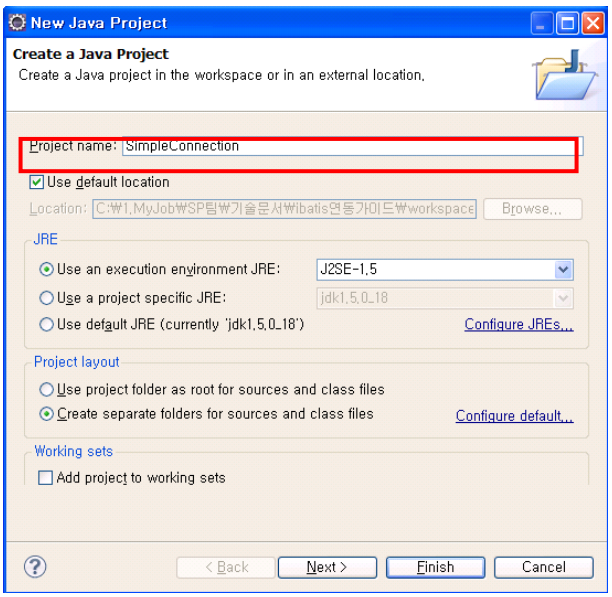
DB에 다음의 테이블과 시퀀스를 생성한다. (create_tbl.sql 파일 참고)

```
CREATE TABLE PERSON(
        PER_ID NUMBER (5, 0) NOT NULL,
        PER_NAME VARCHAR (40) NOT NULL,
        PER_BIRTH_DATE DATE,
        PER_WEIGHT_KG NUMBER (4, 2) NOT NULL,
        PER_HEIGHT_M NUMBER (4, 2) NOT NULL,
        PRIMARY KEY (PER_ID)
);

CREATE SEQUENCE PERSON_SEQ;
```

프로젝트 생성

Eclipse에서 SimpleConnection 이라는 프로젝트를 생성한다.

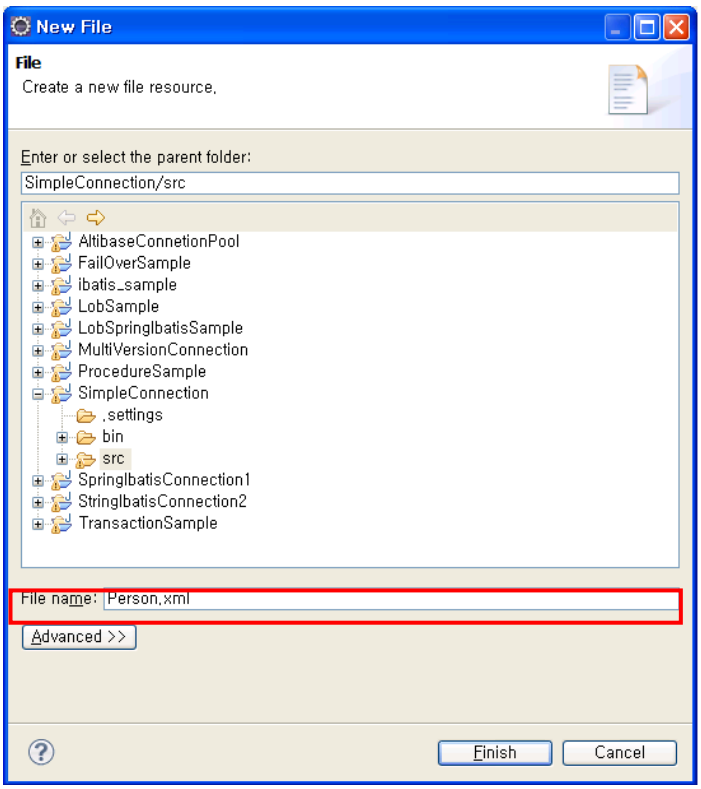
1. 메뉴 – File – Java Project 클릭
2. Project name : 에 SimpleConnection 입력
3. Finish 버튼을 클릭



SqlMap 파일 작성

Person 테이블의 CRUD SQL 구문과 mapping되는 메소드들을 정의한 SqlMap 파일을 작성한다.(Person.xml)

1. SimpleConnection 프로젝트 – src 디렉토리에서 마우스 오른쪽 버튼 클릭하여 New – File을 클릭한다.
2. File name: 에 Person.xml을 작성한다.



다음의 내용을 Person.xml 파일에 작성한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//iBATIS.com//DTD SQL Map 2.0//EN"
"http://www.ibatis.com/dtd/sql-mapconfig-2.dtd">

<sqlMap namespace="Person">
    <resultMap id="PersonResult" class="examples.domain.Person">
            <result property="id" column="per_id" />
            <result property="name" column="per_name" />
            <result property="birthDate" column="per_birth_date" />
            <result property="weightInKilograms" column="per_weight_kg" />
            <result property="heightInMeters" column="per_height_m" />
    </resultMap>

    <select id="getPerson" parameterClass="int"
      resultClass="examples.domain.Person">
            <![CDATA[
                SELECT
                PER_ID as id,
                PER_NAME as name,
                PER_BIRTH_DATE as birthDate,
```

```
                    PER_WEIGHT_KG as weightInKilograms,
                    PER_HEIGHT_M as heightInMeters
                    FROM PERSON
                    WHERE PER_ID = #value#
                ]]>
    </select>

    <insert id="insertPerson" parameterClass="examples.domain.Person">
                <![CDATA[
                    INSERT INTO
                    PERSON (PER_ID, PER_NAME, PER_BIRTH_DATE,
                    PER_WEIGHT_KG, PER_HEIGHT_M)
                    VALUES (#id#, #name#, #birthDate#,
                            #weightInKilograms#, #heightInMeters#)
                ]]>
    </insert>

    <update id="updatePerson" parameterClass="examples.domain.Person">
                <![CDATA[
                    UPDATE PERSON
                    SET PER_NAME = #name#,
                    PER_BIRTH_DATE = #birthDate#,
                    PER_WEIGHT_KG = #weightInKilograms#,
                    PER_HEIGHT_M = #heightInMeters#
                    WHERE PER_ID = #id#
                ]]>
    </update>

    <delete id="deletePerson" parameterClass="int">
                <![CDATA[
                    DELETE PERSON
                    WHERE PER_ID = #id#
                ]]>
    </delete>

    <select id="getAllPersons"    resultMap="PersonResult">
                <![CDATA[
                    SELECT * FROM person
                ]]>
    </select>
</sqlMap>
```

Application에서 SqlMapClient의 insert, update, delete, queryForXXX()메소드를 호출할 때 위의 파일에 정의되어 있는 <insert>, <update>, <delete>, <select> 태그에 정의되어 있는 id와 일치하는 SQL 문들이 자동으로 수행이 된다.

SqlMapConfig 파일 작성

1. ALTIBASE 연결을 위한 property들을 정의한 properties 파일(db.properties)을 작성한다. (SimpleConnection 프로젝트 – src 디렉토리에서 마우스 오른쪽 버튼 클릭하여 New – File을 클릭한다. File name: 에 db.properties을 작성한다.)

```
driver=Altibase.jdbc.driver.AltibaseDriver
url=jdbc:Altibase://192.168.1.35:21129/mydb
username=sys
password=manager
```

2. SqlMapConfig 파일(SqlMapConfigExample.xml)에 ALTIBASE와 연동을 위한 dataSource와 SqlMap 파일을 설정한다. (SimpleConnection 프로젝트 – src 디렉토리에서 마우스 오른쪽 버튼 클릭하여 New – File을 클릭한다. File name: 에 SqlMapConfigExample.xml을 작성한다.)

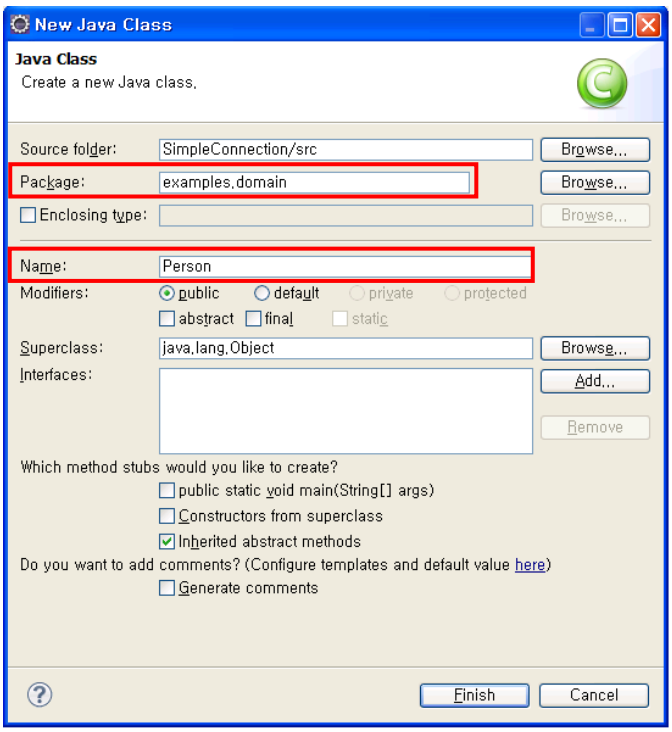
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig PUBLIC "-//iBATIS.com//DTD SQL Map Config
2.0//EN"
"http://www.ibatis.com/dtd/sql-map-config-2.dtd">

<sqlMapConfig>

    <properties resource="db.properties" />

    <transactionManager type="JDBC" >
        <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="${driver}"/>
            <property name="JDBC.ConnectionURL" value="${url}"/>
            <property name="JDBC.Username" value="${username}"/>
            <property name="JDBC.Password" value="${password}"/>
        </dataSource>
    </transactionManager>

    <sqlMap resource="Person.xml" />

</sqlMapConfig>
```

Application 작성

1. person 테이블에 대한 DO객체인 Person 클래스(Person.java)를 작성한다.

   1-1. SimpleConnection 프로젝트의 src 디렉토리에서 마우스 오른쪽 버튼 클릭하여 New – Class를 클릭한다.

   1-2. Package: 에 examples.domain를 입력하고 Name: 에 Person를 입력한다.



다음의 내용을 Person.java 파일에 작성한다.

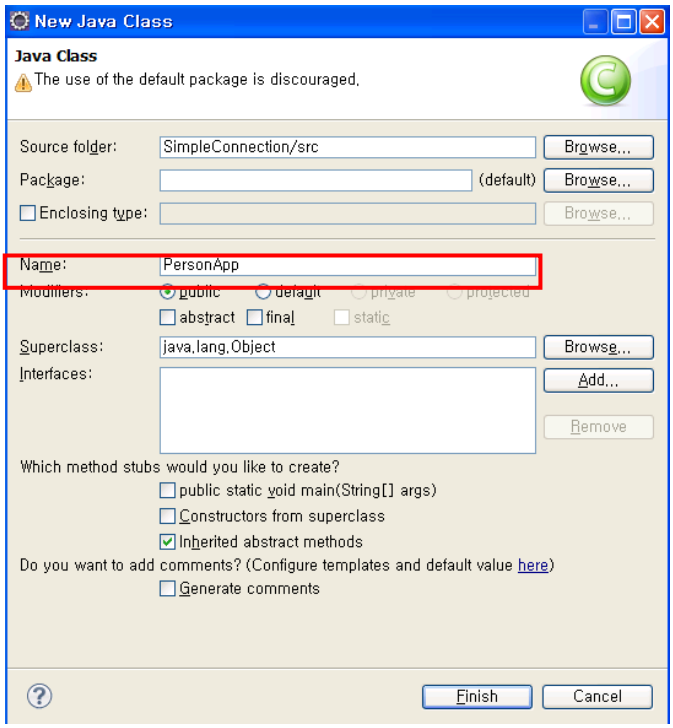
```
package examples.domain;
import java.sql.*;
public class Person {
        private int id;
        private String name;
        private Date birthDate;
        private double weightInKilograms;
        private double heightInMeters;
        public int getId () {
                return id;
        }
        public void setId (int id) {
                this.id = id;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getName() {
                return name;
        }
        public void setBirthDate(Date birthDate) {
                this.birthDate = birthDate;
        }
        public Date getBirthDate() {
                return birthDate;
        }
        public void setWeightInKilograms(double weightInKilograms) {
                this.weightInKilograms = weightInKilograms;
        }
        public double getWeightInKilograms() {
                return weightInKilograms;
        }
```

```
        public void setHeightInMeters(double heightInMeters) {
                this.heightInMeters = heightInMeters;
        }
        public double getHeightInMeters() {
                return heightInMeters;
        }
        public String toString(){
                return "id="+id+", name="+name+", birthData="+birthDate+",
                        weightInKillograms="+weightInKilograms+
                        ",heightInMeters="+heightInMeters;
        }
}
```

2. DB에 CRUD를 실행하는 main 프로그램(PersonApp.java)을 작성한다.

2-1. SimpleConnection 프로젝트의 src 디렉토리에서 마우스 오른쪽 버튼 클릭하여 New – Class를 클릭한다.

2-2. Name: 에 PersonApp를 입력한다.



다음의 내용을 PersonApp.java 파일에 작성한다.

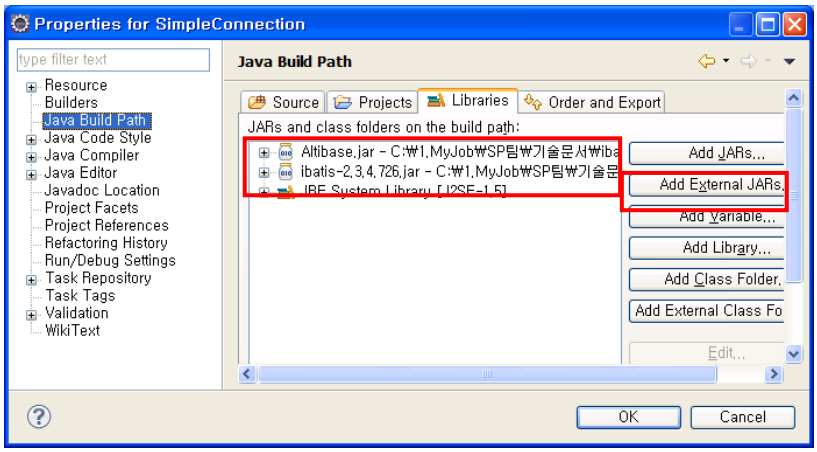
```
import java.io.IOException;
import java.io.Reader;
import java.sql.SQLException;
import java.util.List;
import examples.domain.Person;
import com.ibatis.common.resources.Resources;
import com.ibatis.sqlmap.client.SqlMapClient;
import com.ibatis.sqlmap.client.SqlMapClientBuilder;

public class PersonApp {

    public static void main(String[] args) throws Exception {
        String resource ="SqlMapConfigExample.xml";
```

```
            Reader reader = Resources.getResourceAsReader(resource);
            SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);

            //insert Person
            Person newPerson1 = new Person();
            newPerson1.setName("KIM");
            newPerson1.setBirthDate (new java.sql.Date(1978,1-1,1));
            newPerson1.setHeightInMeters(1.82);
            newPerson1.setWeightInKilograms(80.23);
            sqlMap.insert ("insertPerson", newPerson1);

            Person newPerson2 = new Person();
            newPerson2.setName("LEE");
            newPerson2.setBirthDate (new java.sql.Date(1975,5-1,5));
            newPerson2.setHeightInMeters(1.57);
            newPerson2.setWeightInKilograms(45.23);
            sqlMap.insert ("insertPerson", newPerson2);
            System.out.println();
            System.out.println("insert Person");

            List<Person> list = (List<Person>)sqlMap.queryForList("getAllPersons");
            System.out.println("Selected "+list.size()+" records.");
            for(int i=0; i< list.size();i++){
                    System.out.println(list.get(i));
            }

            //update Person
            newPerson1.setHeightInMeters(1.93);
            newPerson1.setWeightInKilograms(86.36);
            sqlMap.update("updatePerson", newPerson1);
            System.out.println();
            System.out.println("update Person");
            list = sqlMap.queryForList("getAllPersons");
            System.out.println("Selected "+list.size()+" records.");
            for(int i=0; i< list.size();i++){
                    System.out.println(list.get(i).toString());
            }
            System.out.println();
            System.out.println("get Person");

            //get Person
            Integer personPk = new Integer(1);
            Person person = (Person) sqlMap.queryForObject ("getPerson", personPk);
            System.out.println(person);

            //delete Person
            sqlMap.delete ("deletePerson", new Integer(1));
            sqlMap.delete ("deletePerson", new Integer(2));
            System.out.println();
            System.out.println("delete Person");
            list = sqlMap.queryForList("getAllPersons");
            System.out.println("Selected "+list.size()+" records.");
            for(int i=0; i< list.size();i++){
                    System.out.println(list.get(i));
            }

    }
}
```

관련 JAR 파일 추가

Altibase.jar와 ibatis-2.3.4.x.jar 파일을 추가한다.

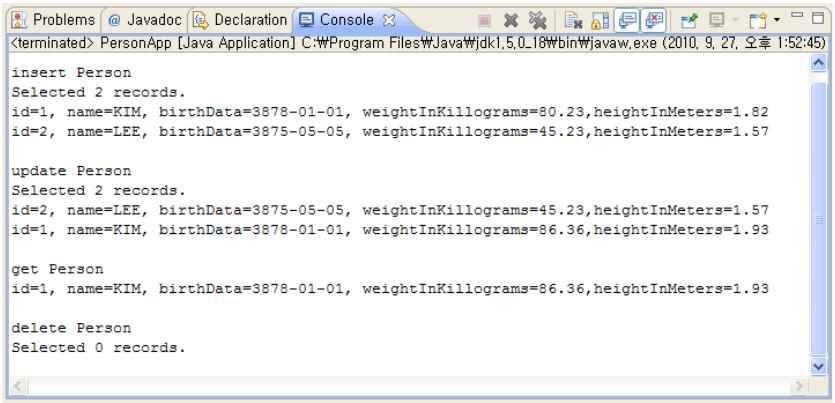
SimpleConnection 프로젝트에서 마우스 오른쪽 버튼 클릭하여 Properties를 클릭 – Java Build Path – Libraries 에서 Add External JARS를 클릭하여 Altibase.jar와 ibatis-2.3.4.x.jar 파일을 추가한다.



Application 실행

SimpleConnection 프로젝트를 실행한다.

SimpleConnection 프로젝트를 클릭한 후 메뉴에서 Run을 실행하거나 Run 실행 단추를 클릭한다.



알티베이스㈜
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 1008 호
02-2082-1000
http://www.altibase.com

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

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

솔루션센터
02-2082-1114
http://support.altibase.com