iBATIS SQL Maps 튜토리얼

For SQL Maps Version 2.0

June 17, 2004

번역 : 이동국(fromm0@gmail.com)
오타 및 오역은 위 메일주소로 보내주시기 바랍니다.

소개

이 튜토리얼은 SQL Maps의 전형적인 사용법을 설명한다. 각각의 주제에 대해서 좀더 상세한 정보는 http://www.ibatis.com 에서 얻을수 있는 SQL Maps개발자가이드를 통해서 찾아볼 수 있다.

SQL Maps 을 사용하기 위해 준비하기

SQL Maps프레임워크는 나쁜 데이터베이스모델과 객체모델에 매우 관대하다. 이것에도 불구하고 당신의 데이터베이스(또는 정규화)와 객체를 디자인할 때 가장 좋은 예제를 사용하도록 추천한다. 다음과 같이 함으로써 당신은 좋은 성능과 깔끔한 디자인을 얻을 수 있을 것이다.

시작하기 위해 가장 쉬운 방법은 당신이 수행하고자 하는 것을 분석하는 것이다. 당신의 비즈니스 객체는 무엇인가.? 당신의 데이터베이스 테이블은 무엇인가.? 그들은 각각 어떻게 관계되는가.? 첫 예제를 위해 전형적인 자바빈즈 플랫폼을 위해 다음의 간단한 Person클래스를 보자.

Person.java

```
package examples.domain;

//imports implied….

public class Person {

   private int id;
   private String firstName;
   private String lastName;
   private Date birthDate;
   private double weightInKilograms;
   private double heightInMeters;

   public int getId () {
      return id;
   }
   public void setId (int id) {
      this.id = id;
   }

   //…let's assume we have the other getters and setters to save space…
}
```

우리의 데이터베이스에 이 Person클래스를 어떻게 맵핑 시킬까.? SQL Maps는 테이블 대 클래스, 다중 테이블 대 클래스, 다중 클래스 대 테이블등과 같은 관계로부터 어떠한 제약도 가지지 않는다. 당신이 유효한 SQL의 모든 기능을 가지기 때문에 제약은 거의 없다. 이 예제를 위해 테이블 대 클래스 관계를 나타내기에 적합한 다음의 간단한 테이블을 사용하자.

Person.sql

```
CREATE TABLE PERSON(
   PER_ID NUMBER (5, 0) NOT NULL,
   PER_FIRST_NAME VARCHAR (40) NOT NULL,
   PER_LAST_NAME VARCHAR (40) NOT NULL,
   PER_BIRTH_DATE DATETIME ,
   PER_WEIGHT_KG NUMBER (4, 2) NOT NULL,
```

```
  PER_HEIGHT_M NUMBER (4, 2) NOT NULL,
  PRIMARY KEY (PER_ID)
)
```

SQL Map 설정 파일

클래스와 테이블로 작업하기 편안한 시점에 시작하기 가장 좋은 방법은 SQL Map설정 파일이다. 이 파일은 우리의 SQL Map 구현물을 위한 가장 상위 설정으로 작동할 것이다.

이 설정파일은 XML파일이다. 이것은 프라퍼티, JDBC데이터소스 그리고 SQL Maps를 설정할 것이다. 이것은 당신에게 많은 수의 다른 형태로 구현될 수 있는 데이터소스를 중앙집중적으로 설정할 수 있는 편리한 위치를 준다. 프레임워크는 iBATIS SimpleDataSource, Jakarta DBCP, JNDI를 검색을 통해 찾을 수 있는 어떤 데이터소스를 포함해서 많은 수의 데이터소스 구현물을 다룰 수 있다. 개발자 가이드에서 좀더 상세하게 다루어진다. 구조는 아래 예제처럼 간단하다.

SqlMapConfigExample.xml

```
<?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">

<!--Always ensure to use the correct XML header as above! -->

<sqlMapConfig>

 <!--The properties (name=value) in the file specified here can be used placeholders in this config file (e.g.
     "${driver}". The file is usually relative to the classpath and is optional. -->

 <properties resource="examples/sqlmap/maps/SqlMapConfigExample.properties" />

 <!--These settings control SqlMap configuration details, primarily to do with transaction
     management. They are all optional (see the Developer Guide for more). -->


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

 />

 <!--Type aliases allow you to use a shorter name for long fully qualified class names. -->

 <typeAlias alias="order" type="testdomain.Order"/>

 <!--Configure a datasource to use with this SQL Map using SimpleDataSource.
     Notice the use of the properties from the above resource -->


 <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>


 <!--Identify all SQL Map XML files to be loaded by this SQL map. Notice the paths
      are relative to the classpath. For now, we only have one... -->


 <sqlMap resource="examples/sqlmap/maps/Person.xml" />

</sqlMapConfig>
```

SqlMapConfigExample.properties

```
# This is just a simple properties file that simplifies automated configuration
# of the SQL Maps configuration file (e.g. by Ant builds or continuous
# integration tools for different environments... etc.)
# These values can be used in any property value in the file above (e.g. "${driver}")
# Using a properties file such as this is completely optional.


driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:oracle1
username=jsmith
password=test
```

SQL Map 파일들

지금 우리는 설정된 데이터소스를 가지고 있고 중앙집중적 설정파일은 수행할 준비가 되었다. 우리는 SQL코드와 파라미터객체를 위한 맵핑 그리고 result객체를 포함하는 SQL Map파일을 제공할 필요가 있을 것이다.

아래의 예제를 가지고 계속적으로 Person클래스와 PERSON테이블을 위해 SQL Map파일을 빌드해보자. 우리는 SQL문서를 일반적인 구조와 간단한 select문을 가지고 시작할 것이다.

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-map-
2.dtd">
<sqlMap namespace="Person">
  <select id="getPerson" resultClass="examples.domain.Person">
    SELECT PER_ID as id,
      PER_FIRST_NAME as firstName,
      PER_LAST_NAME as lastName,
      PER_BIRTH_DATE as birthDate,
      PER_WEIGHT_KG as weightInKilograms,
      PER_HEIGHT_M as heightInMeters
    FROM PERSON
    WHERE PER_ID = #value#
  </select>
</sqlMap>
```

위의 예제는 SQL Map의 가장 간단한 형태를 보여준다. 이것은 이름매치(name matching)에 기초로 하여 ResultSet의 칼럼을 자바빈즈 프라퍼티(또는 map키 등등)에 자동적으로 맵핑하는 SQL Maps프레임워크의 기능을 사용한다. #value# 토큰은 입력 파라미터이다. 좀더 상세하게 말하면 "value"의 사용은 우리가 간단한 원시래퍼타입(이를 테면, Integer 하지만 우리는 이것에 어떠한 제한도 하지 않는다.)을 사용하는 것을 적용한다.

매우 간단함에도 불구하고 자동결과맵핑(auto-result mapping) 접근법을 사용하는 데에는 몇 가지 제한이 있다. 출력 칼럼의 타입을 정의하거나 관계된 데이터(복잡한 프라퍼티)를 자동적으로 로드하는 방법이 없다. 그리고 이 접근법은 ResultSetMetaData을 접근하는 것을 요구하기에 미세한 성능 영향이 있다. resultMap을 사용함으로써 우리는 이런 제한점 모두를 극복할 수 있다. 하지만 지금 단순함은 우리의 목표이다. 그리고 우리는 나중에 다른 접근법으로 변경할 수도(자바소스코드에 어떠한 변경도 없이) 있다.
대개의 데이터베이스 애플리케이션은 데이터베이스로부터 간단하게 읽지 않는다. 그들은 데이터베이스내 데이터를 변경할 수도 있습니다. 우리는 맵핑된 statement내 SELECT를 간단히 찾을 수 있는 예제를 이미 보았습니다. 하지만 INSERT, UPDATE 그리고 DELETE에 대해서는.? 좋은 뉴스는 전혀 다르지 않다는 것이다. 아래에서 우리는 데이터에 접근하고 변경하기 위한 완벽한 statement을 제공하기 위해 좀더 많은 statement으로 우리의 Person SQL Map를 완성할것이다.

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-map-
  2.dtd">

<sqlMap namespace="Person">

   <!--Use primitive wrapper type (e.g. Integer) as parameter and allow results to
      be auto-mapped results to Person object (JavaBean) properties -->

   <select id="getPerson" parameterClass="int" resultClass="examples.domain.Person">
        SELECT
                PER_ID as id,
                PER_FIRST_NAME as firstName,
                PER_LAST_NAME as lastName,
                PER_BIRTH_DATE as birthDate,
                PER_WEIGHT_KG as weightInKilograms,
                PER_HEIGHT_M as heightInMeters

  FROM PERSON
  WHERE PER_ID = #value#
  </select>


   <!--Use Person object (JavaBean) properties as parameters for insert. Each of the parameters in the
      #hash# symbols is a JavaBeans property. -->
   <insert id="insertPerson" parameterClass="examples.domain.Person">
        INSERT INTO
        PERSON (PER_ID, PER_FIRST_NAME, PER_LAST_NAME,

                PER_BIRTH_DATE, PER_WEIGHT_KG, PER_HEIGHT_M)
     VALUES (#id#, #firstName#, #lastName#,
     #birthDate#, #weightInKilograms#, #heightInMeters#)
     </insert>


   <!--Use Person object (JavaBean) properties as parameters for update. Each of the parameters in the
      #hash# symbols is a JavaBeans property. -->
```

```
    <update id="updatePerson" parameterClass="examples.domain.Person">
        UPDATE PERSON
        SET PER_FIRST_NAME = #firstName#,

            PER_LAST_NAME = #lastName#, PER_BIRTH_DATE = #birthDate#,
            PER_WEIGHT_KG = #weightInKilograms#,
            PER_HEIGHT_M = #heightInMeters#

        WHERE PER_ID = #id#
    </update>


    <!--Use Person object (JavaBean) "id" properties as parameters for delete. Each of the parameters in the
        #hash# symbols is a JavaBeans property. -->
    <delete id="deletePerson" parameterClass="examples.domain.Person">
        DELETE PERSON
        WHERE PER_ID = #id#

    </delete>

</sqlMap>
```

SQL Map 프레임워크로 프로그래밍하기.

지금 그것은 우리가 모두 설정하고 맵핑했다. 우리가 해야 할 필요가 있는 모든 것은 자바애플리케이션내의 코드이다. 첫번째 단계는 SQL Map를 설정하는 것이다. 이것은 우리가 이전에 생성한 SQL Map 설정 XML파일을 로드하는 간단한 방법이다. XML파일 로딩을 간단히 하기 위해서 우리는 프레임워크와 함께 포함된 Resource클래스를 사용할 수 있다.

```
String resource = "com/ibatis/example/sqlMap-config.xml";
Reader reader = Resources.getResourceAsReader (resource);
SqlMapClient sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
```

SqlMapClient 객체는 오랜 기간 지속되고 쓰레드에 안전한(thread safe) 서비스 객체이다. 주어진 애플리케이션 실행을 위해 당신은 오직 한번 그것을 초기화하고 설정할 필요가 있다. 이것은 정적 멤버와 기본 클래스를 위해 좋은 지원자를 만들어 주거나 당신이 좀더 중앙집중적으로 설정하고 전역적으로 유용하게 만들기를 선호한다면 당신은 당신 자신의 편리한 클래스를 만들 수 있다. 여기에 당신이 쓸 수 있는 편리한 클래스 예제가 있다.

```
public MyAppSqlConfig {

  private static final SqlMapClient sqlMap;

  static {
    try {
      String resource = "com/ibatis/example/sqlMap-config.xml";
      Reader reader = Resources.getResourceAsReader (resource);
      sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);

    } catch (Exception e) {
      // If you get an error at this point, it doesn't matter what it was. It is going to be //
      unrecoverable and we will want the app to blow up hard so we are aware of the // problem.
      You should always log such errors and re-throw them in such a way that // you can be made
      immediately aware of the problem.
  e.printStackTrace();
  throw new RuntimeException ("Error initializing
  MyAppSqlConfig class. Cause: " + e);
```

```
    }
}

public static getSqlMapInstance () {
    return sqlMap;
    }

}
```

데이터베이스로부터 객체 읽기

지금 SqlMap인스턴스는 초기화되었고 쉽게 접근가능하다. 우리는 이것을 사용할 수 있다. 데이터베이스로부터 Person객체를 얻기 위해 이것을 사용해 보자.

데이터베이스로부터 Person객체를 얻기 위해 우리는 간단하게 SqlMap인스턴스, 맵핑된 statement의 이름 그리고 Person ID가 필요하다. Person #5 를 로드해보자.

```
...
SqlMapClient sqlMap = MyAppSqlMapConfig.getSqlMapInstance();
// as coded above ...
Integer personPk = new Integer(5);
Person person = (Person) sqlMap.queryForObject ("getPerson", personPk);
...
```

데이터베이스에 객체 쓰기

우리는 지금 데이터베이스로부터 Person객체를 가진다. 몇몇 데이터를 변경해보자. 우리는 person의 height와 weight를 변경할 것이다.

```
...
person.setHeightInMeters(1.83);
// person as read above
person.setWeightInKilograms(86.36);
...
sqlMap.update("updatePerson", person);
...
```

만약 우리가 이 Person을 삭제하기를 원한다면 다음처럼 쉽다.

```
...
sqlMap.delete ("deletePerson", person);
...
```

새로운 Person을 추가하는 것은 유사하다.

```
Person newPerson = new Person();
newPerson.setId(11); // you would normally get the ID from a sequence or custom table
newPerson.setFirstName("Clinton");
newPerson.setLastName("Begin");
newPerson.setBirthDate (null);
newPerson.setHeightInMeters(1.83);
newPerson.setWeightInKilograms(86.36);
...
sqlMap.insert ("insertPerson", newPerson);
...
```

모든 것을 수행했다.!!

다음 단계 ...

이것은 짧은 튜토리얼의 끝이다. 완벽한 SQL Maps 2.0개발자 가이드를 위해서는 http://www.ibatis.com 를 방문하길 바란다. Jakarta Struts, iBATIS DAO 2.0 그리고 SQL Maps 2.0을 기반으로 하는 완벽한 웹애플리케이션 예지인 JPetStore 4또한 제공한다.