iBATIS SQL Maps

入门教程

Version 2.0

2004 年 6 月 17 日

Clinton Begin 著

刘 涛 译

简介

本文是初学者的快速入门教程，涵盖了 SQL Map 的一个简单而典型的应用。每个主题更详细的信息可以参考《iBatis SQL Maps 2.0 开发指南》。

本文是《iBatis SQL Maps Tutorial》的中文版，仅供读者参考。最权威的以 Clinton Begin 的官方文档为准，它可以从 http://www.ibatis.com 网站下载。如果中文翻译有错误，请通知译者（email：toleu@21cn.com，Blog：http://starrynight.blogdriver.com/）。

准备使用 SQL Map

SQL Map 架构能应用于设计不好的数据库模型甚至是设计不好的对象模型。尽管如此，您在设计数据库模型和对象模型时，还是应该遵循最佳的设计原则。这样，您会获得更好的性能和更简洁清晰的设计方案。

设计最容易开始的地方是分析应用的业务逻辑。分析什么是应用的业务对象，什么是数据模型以及两者之间的关系。作为快速入门第一个例子，我们使用一个简单的 Java Bean Person 类。

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…
}
```

Person 类有了，如何将 Person 类映射成数据表呢？SQL Map 对 Java Bean 和数据表之间的关系没有限制，如一个数据表映射成一个 Java Bean，或多个表映射成一个 Java Bean，或多个 Java Bean 映射成一个数据表等。因为使用 SQL Map 您可以充分发挥 SQL 语句的全部潜力而很少限制。下面这个例子，我们使用一个简单的表，将一个表映射成一个 Java Bean，Java Bean 和表是一对一的关系。

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 的配置文件

现在准备好了学习环境，让我们从学习 SQL Map 的配置文件开始，配置文件是 SQL MAP 的配置信息统一设置的地方。

SQL Map 配置文件是 XML 文件，我们可以它设置各种属性，JDBC DataSource 和 SQL Map。在配置文件中，可以方便地统一配置 DataSource 不同的实现。SQL Map 框架包括 DataSource 的 iBATIS 实现：SimpleDataSource 类，Jakarta DBCP（Commons），和可通过 JNDI 上下文查找的 DataSource（即应用服务器中的 DataSource）。详细的使用方法在以后的章节讨论。在本例中，我们使用 Jakarta DBCP。对于上面的例子，配置非常简单，如下所示：

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 relative to the classpath and is completely 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 的映射文件

现在 DataSource 已经配置好了，并且有了统一的 SQL Map 配置文件，我们还需要 SQL Map 的映射文件。映射文件包括 SQL 语句和参数对象和结果对象的映射。

继续上面的例子，我们从一个简单的查询语句开始，为 Person 类和 PERSON 表之间创建一个 SQL Map 映射文件。

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>
```

上面的例子是 SQL Map 最简单的形式。它使用了 SQL Map 框架中一个特性，根据匹配的名字将 ResultSet 的列映射成 Java Bean 的属性（或 Map 的 key 值）。#value#符号是输入参数，该符号表示使用了基本类型的包装类作为输入参数（即 Integer，但不仅限于此类型）。

以上的方法虽然很简单，但有一些限制，无法指定输出字段的数据类型，无法自动地在结果对象中载入相关的信息（即 Java Bean 无法使用复杂的属性）；以上的方法对性能还有轻微的不利影响，因为需要读取 ResultSetMetaData 的信息。使用 resultMap，可以克服以上的不足，但现在只需要一个简单的例子，以后我们再转向其他不同的方法（无须修改 Java 代码）。

大多数的应用不仅需要从数据库中读取数据，还需要修改数据。我们已有了一个 SELECT 查询语句的 mapped statement 简单例子，下面看看 INSERT，UPDATE 和 DELETE 的 mapped statement 什么样子。幸运的是，它们其实没什么区别。接下来，我们完成 Person SQL Map 其他部分，以实现修改数据的功能。

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 (Java Bean) 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 (Java Bean) properties as parameters for insert. Each of the
    parameters in the #hash# symbols is a Java Beans 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 (Java Bean) properties as parameters for update. Each of the
    parameters in the #hash# symbols is a Java Beans 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 (Java Bean) "id" properties as parameters for delete. Each of the
    parameters in the #hash# symbols is a Java Beans property. -->
    <delete id="deletePerson" parameterClass="examples.domain.Person">
        DELETE PERSON
        WHERE PER_ID = #id#
    </delete>
</sqlMap>
```

使用 SQL Map 框架编程

好了，我们完成了所有的配置文件和映射文件，就剩下的应用的编码工作了。首先要设置 SQL Map，读入刚创建好的 SQL Map XML 配置文件。为简化这个工作，可以使用 SQL Map 架构中提供的 Resources 类。

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

以上的 SqlMapClient 对象是线程安全，并且应持久生存。对于一个特定的应用，只需进行一次 SqlMap 配置。因此，它可以作为基类的一个静态对象（即 DAO 对象的基类），或者，如果您想让它有更大的作用范围，可以把它封装在方便使用的类中。例如：

```
public class MyAppSqlConfig {
  private static final SqlMapClient sqlMap;
  static {
      try {
          String resource = "com/ibatis/example/sql-map-config.xml";
          Reader reader = Resources.getResourceAsReader (resource);
          sqlMap = SqlMapClientBuilder.buildSqlMapClient(reader);
```

```
        } catch (Exception e) {
            // If you get an error at this point, it matters little what it was. It is going to be
            // unrecoverable and we will want the app to blow up good 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;
    }
}
```

从数据库读取对象

既然 SqlMap 对象已完成初始化，就可以方便地使用它了。首先，我们用它从数据库中读取一个 Person 对象。（在本例中，假设 PERSON 表中已存在 10 条记录，PER_ID 从 1 到 10）。

要从数据库中得到一个 Person 对象，只需要 SqlMap 实例，mapped statement 的名字和一个 Person ID 号。让我们读入 PER_ID 是 5 的 Person 对象。

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

把对象写入数据库

现在已有了一个从数据库中读出的 Person 对象，接着修改 Person 对象的 height 和 weight 属性，并将它写入数据库。

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

要删除这个 Person 对象，也很容易。

```
…
sqlMap.delete(“deletePerson”, person);
…
```

类似地，也可以创建一个新的 Person 对象。

```
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);
…
```

好了，快速入门课程终于学完了。

下一步…

至此本教程结束了。请访问 http://www.ibatis.com 网站下载完整的《iBatis SQL Maps 开发指南》，还有 JPetStore 4，它是一个完整的 Web 应用例子，基于 Jakarta Struts，iBatis DAO 2.0 和 SQL Maps 2.0。

附录：容易出错的地方

本附录是译者添加的，列出了初学者容易出错的地方，作为完成快速入门课程后的学习笔记，可以让初学者少走些弯路。仅供参考。

1) 在 parameterMap 和 resultMap 中，字段数据类型是 java.sql.Types 类定义的常量名称。常用的数据类型包括 BLOB，CHAR，CLOB，DATE，LONGVARBINARY，INTEGER，NULL，NUMERIC，TIME，TIMESTAMP 和 VARCHAR 等。
2) 对于数据表中 NULLBALE 的字段，必须在 parameterMap 和 resultMap 中指定字段的数据类型。
3) 对于数据类型是 DATE，CLOB 或 BLOB 的字段，最好在 parameterMap 和 resultMap

中指定数据类型。

4) 对于二进制类型的数据，可以将 LONGVARBINARY 映射成 byte[]。

5) 对于文本类型较大的数据，可以将 CLOB 映射成 String。

6) Java Bean 必须拥有缺省的构造器（即无参数的构造器）。

7) Java Bean 最好实现 Serializable 接口，以备应用的进一步扩展。