**iBATIS介绍，iBATIS是什么？**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**1**

[下一篇](http://www.yiibai.com/ibatis/ibatis_environment.html)

iBATIS的是一个持久层框架，它能够自动在 [Java](http://www.yiibai.com/java/), .NET, 和Ruby on Rails中与SQL数据库和对象之间的映射。映射是从应用程序逻辑封装在XML配置文件中的SQL语句脱钩。

iBATIS是一个轻量级的框架和持久性API适合持久化的POJO（普通Java对象）。

iBATIS是被称为一个数据映射和映射需要的类的属性和数据库中的表的列之间的参数和结果。

iBATIS和其他持久化框架，如Hibernate之间的显著区别在于，iBATIS强调使用SQL，而其他的框架通常使用一个自定义的查询语言，具有Hibernate查询语言（HQL）或Enterprise JavaBeans的查询语言（EJB QL）。

**iBATIS的设计理念：**

iBatis提供了以下的设计理念：

* 简单: iBATIS的被广泛认为是可用的最简单的持久化框架之一。
* 快速开发：iBATIS的理念是尽一切可能，以方便超快速开发。
* 可移植性: iBATIS可用于几乎任何语言或平台，如Java，Ruby和C＃，微软.NET实现。
* 独立的接口：iBATIS提供独立于数据库的接口和API，帮助应用程序的其余部分保持独立的任何持久性相关的资源，
* 开源：iBATIS是自由和开放源码软件。

**IBATIS的优点**

下面是使用iBATIS的一些优势：

* 支持存储过程：iBATIS的SQL封装以存储过程的形式，使业务逻辑保持在数据库之外，应用程序更易于部署和测试，更便于移植。
* 支持内嵌的SQL：预编译器不是必需的，并有完全访问所有的SQL语句的特性。
* 支持动态SQL: iBATIS特性提供基于参数动态生成SQL查询。
* 支持O / RM：iBATIS支持许多相同的功能作为一个O / RM工具，如延迟加载，连接抓取，缓存，运行时代码生成和继承

**先决条件：**

在开始之前，要确保你了解过程和面向对象编程的基本知识：控制结构，数据结构和变量，类，对象等。

iBATIS使用Java编程语言开发面向数据库应用程序。

要理解Java编程细节可以通过我们的[Java教程](http://www.yiibai.com/java/index.html)。

**iBATIS配置环境**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**1**

[上一篇](http://www.yiibai.com/ibatis/ibatis_overview.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_create_operation.html)

在开始使用iBATIS开发之前，必须设置你的环境正常。本教程将指导您用几个步骤来实现的工作环境。

**iBATIS 安装:**

这里有几个简单的步骤，需要开展Linux机器上安装iBATIS：

* 下载iBATIS的最新版本 [下载iBATIS](http://ibatis.apache.org/java.cgi).
* 解压下载的文件，从包中提取.jar文件并将其保存在相应的lib目录下。
* 在提取 .jar文件适当设置PATH和CLASSPATH变量。

下面是进行Linux机器下载iBATIS的二进制文件的步骤：

$ unzip ibatis-2.3.4.726.zip

inflating: META-INF/MANIFEST.MF

creating: doc/

creating: lib/

creating: simple\_example/

creating: simple\_example/com/

creating: simple\_example/com/mydomain/

creating: simple\_example/com/mydomain/data/

creating: simple\_example/com/mydomain/domain/

creating: src/

inflating: doc/dev-javadoc.zip

inflating: doc/user-javadoc.zip

inflating: jar-dependencies.txt

inflating: lib/ibatis-2.3.4.726.jar

inflating: license.txt

inflating: notice.txt

inflating: release.txt

$pwd

/var/home/ibatis

$set PATH=$PATH:/var/home/ibatis/

$set CLASSPATH=$CLASSPATH:/var/home/ibatis\

/lib/ibatis-2.3.4.726.jar

**数据库设置：**

使用下面的语法在 MySQL数据库中创建EMPLOYEE表：

mysql> CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

**创建SqlMapConfig.xml**

考虑以下几点：

* 我们将使用JDBC来访问数据库 testdb.
* MySQL的JDBC驱动程序是 "com.mysql.jdbc.Driver".
* 连接URL是 "jdbc:mysql://localhost:3306/testdb".
* 使用的用户名和密码是 "root" and "root".
* SQL语句映射的所有操作将被描述在"Employee.xml".

基于上述假设，我们必须创建一个XML配置文件，nameSqlMapConfig.xml以下内容。这就是需要提供所需的iBatis的所有配置：

这两个文件SqlMapConfig.xml和Employee.xml 存在于类路径。现在，我们将保持Employee.xml文件为空，我们将格式转换的在随后的章节内容。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMapConfig

PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>

<settings useStatementNamespaces="true"/>

<transactionManager type="JDBC">

<dataSource type="SIMPLE">

<property name="JDBC.Driver"

value="com.mysql.jdbc.Driver"/>

<property name="JDBC.ConnectionURL"

value="jdbc:mysql://localhost:3306/testdb"/>

<property name="JDBC.Username" value="root"/>

<property name="JDBC.Password" value="root"/>

</dataSource>

</transactionManager>

<sqlMap resource="Employee.xml"/>

</sqlMapConfig>

还有其他一些可选的属性，您可以在SqlMapConfig.xml文件中设置：

<property name="JDBC.AutoCommit" value="true"/>

<property name="Pool.MaximumActiveConnections" value="10"/>

<property name="Pool.MaximumIdleConnections" value="5"/>

<property name="Pool.MaximumCheckoutTime" value="150000"/>

<property name="Pool.MaximumTimeToWait" value="500"/>

<property name="Pool.PingQuery" value="select 1 from Employee"/>

<property name="Pool.PingEnabled" value="false"/>

**iBATIS创建操作**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**0**

[上一篇](http://www.yiibai.com/ibatis/ibatis_environment.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_read_operation.html)

若要使用iBATIS执行的任何CRUD（创建，写入，更新和删除）操作，需要创建一个的POJO（普通Java对象）类对应的表。本课程介绍的对象，将“模式”的数据库表中的行。

POJO类必须实现所有执行所需的操作所需的方法。

我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

**Employee POJO 类：**

我们会在Employee.java文件中创建Employee类，如下所示：

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

} /\* End of Employee \*/

可以定义方法来设置表中的各个字段。下一章节将告诉你如何获得各个字段的值。

**Employee.xml 文件:**

要定义使用iBATIS SQL映射语句中，我们将使用<insert>标签，这个标签定义中，我们会定义将用于在IbatisInsert.java文件的数据库执行SQL INSERT查询“id”。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<insert id="insert" parameterClass="Employee">

insert into EMPLOYEE(first\_name, last\_name, salary)

values (#first\_name#, #last\_name#, #salary#)

<selectKey resultClass="int" keyProperty="id">

select last\_insert\_id() as id

</selectKey>

</insert>

</sqlMap>

这里parameterClass：可以采取一个值作为字符串，整型，浮点型，double或根据要求任何类的对象。在这个例子中，我们将通过Employee对象作为参数而调用SqlMap类的insert方法。

如果您的数据库表使用IDENTITY，AUTO\_INCREMENT或串行列或已定义的SEQUENCE/GENERATOR，可以使用<selectKey>元素在的<insert>语句中使用或返回数据库生成的值。

**IbatisInsert.java 文件:**

文件将应用程序级别的逻辑在Employee表中插入记录：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisInsert{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

/\* This would insert one record in Employee table. \*/

System.out.println("Going to insert record.....");

Employee em = new Employee("Zara", "Ali", 5000);

smc.insert("Employee.insert", em);

System.out.println("Record Inserted Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述软件。请确保已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisInsert.java如上图所示，并编译它。
* 执行IbatisInsert二进制文件来运行程序。

会得到下面的结果，并创纪录的将在EMPLOYEE表中创建。

$java IbatisInsert

Going to insert record.....

Record Inserted Successfully

去查看EMPLOYEE表，它应该有如下结果：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

+----+------------+-----------+--------+

1 row in set (0.00 sec)

**iBATIS读取操作**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**0**

[上一篇](http://www.yiibai.com/ibatis/ibatis_create_operation.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_update_operation.html)

上一章展示了如何使用iBATIS执行创建操作表。本章将告诉你如何使用iBATIS来读取表。

我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

此表有如下只有一条记录：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

+----+------------+-----------+--------+

1 row in set (0.00 sec)

**Employee POJO 类:**

要执行读操作，我们将修改Employee类中Employee.java文件，如下所示：

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

/\* Here are the method definitions \*/

public int getId() {

return id;

}

public String getFirstName() {

return first\_name;

}

public String getLastName() {

return last\_name;

}

public int getSalary() {

return salary;

}

} /\* End of Employee \*/

**Employee.xml 文件:**

要定义使用iBATIS SQL映射语句，我们将增加<select>标记在Employee.xml文件，这个标签定义中，我们会定义将用于在IbatisRead.java文件的数据库执行SQL SELECT查询的“id”。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<insert id="insert" parameterClass="Employee">

INSERT INTO EMPLOYEE(first\_name, last\_name, salary)

values (#first\_name#, #last\_name#, #salary#)

<selectKey resultClass="int" keyProperty="id">

select last\_insert\_id() as id

</selectKey>

</insert>

<select id="getAll" resultClass="Employee">

SELECT \* FROM EMPLOYEE

</select>

</sqlMap>

在这里，我们没有使用WHERE子句和SQL SELECT语句。后续章节将演示如何用WHERE和SELECT语句子句，以及如何将值传递到WHERE子句。

**IbatisRead.java 文件:**

该文件将应用程序级别的逻辑从雇员Employee表中读出记录：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisRead{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

/\* This would read all records from the Employee table. \*/

System.out.println("Going to read records.....");

List <Employee> ems = (List<Employee>)

smc.queryForList("Employee.getAll", null);

Employee em = null;

for (Employee e : ems) {

System.out.print(" " + e.getId());

System.out.print(" " + e.getFirstName());

System.out.print(" " + e.getLastName());

System.out.print(" " + e.getSalary());

em = e;

System.out.println("");

}

System.out.println("Records Read Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述应用。请确保您已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisRead.java如上图所示，并编译它。
* 执行IbatisRead二进制文件来运行程序。

你会得到下面的结果，并且将记录从EMPLOYEE表中读取。

Going to read records.....

1 Zara Ali 5000

Record Reads Successfully

**iBATIS更新操作**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**1**

[上一篇](http://www.yiibai.com/ibatis/ibatis_read_operation.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_delete_operation.html)

上一章展示了如何使用iBATIS对表进行读取操作。本章将告诉你如何在一个表中使用iBATIS更新记录。

我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

此表有如下只有一条记录：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

+----+------------+-----------+--------+

1 row in set (0.00 sec)

**Employee POJO 类:**

要执行UDPATE操作，需要修改Employee.java文件，如下所示：

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

/\* Here are the required method definitions \*/

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return first\_name;

}

public void setFirstName(String fname) {

this.first\_name = fname;

}

public String getLastName() {

return last\_name;

}

public void setlastName(String lname) {

this.last\_name = lname;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

} /\* End of Employee \*/

**Employee.xml 文件:**

要定义使用iBATIS SQL映射语句，我们想补充的<Update>标签Employee.xml文件，这个标签定义中，我们会定义将用于在IbatisUpdate.java文件的数据库执行SQL UPDATE查询的“id”。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<insert id="insert" parameterClass="Employee">

INSERT INTO EMPLOYEE(first\_name, last\_name, salary)

values (#first\_name#, #last\_name#, #salary#)

<selectKey resultClass="int" keyProperty="id">

select last\_insert\_id() as id

</selectKey>

</insert>

<select id="getAll" resultClass="Employee">

SELECT \* FROM EMPLOYEE

</select>

<update id="update" parameterClass="Employee">

UPDATE EMPLOYEE

SET first\_name = #first\_name#

WHERE id = #id#

</update>

</sqlMap>

**IbatisUpdate.java 文件:**

文件将应用程序级别的逻辑来更新记录到Employee表：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisUpdate{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

/\* This would update one record in Employee table. \*/

System.out.println("Going to update record.....");

Employee rec = new Employee();

rec.setId(1);

rec.setFirstName( "Roma");

smc.update("Employee.update", rec );

System.out.println("Record updated Successfully ");

System.out.println("Going to read records.....");

List <Employee> ems = (List<Employee>)

smc.queryForList("Employee.getAll", null);

Employee em = null;

for (Employee e : ems) {

System.out.print(" " + e.getId());

System.out.print(" " + e.getFirstName());

System.out.print(" " + e.getLastName());

System.out.print(" " + e.getSalary());

em = e;

System.out.println("");

}

System.out.println("Records Read Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述软件。请确保您已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisUpdate.java如上图所示，并编译它。
* 执行IbatisUpdate二进制文件来运行程序。

得到下面的结果，并创建纪录在EMPLOYEE表进行更新和更高版本相同的记录将从EMPLOYEE表中读出。

Going to update record.....

Record updated Successfully

Going to read records.....

1 Roma Ali 5000

Records Read Successfully

**iBATIS删除操作**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**0**

[上一篇](http://www.yiibai.com/ibatis/ibatis_update_operation.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_result_maps.html)

本章将教你如何从表中使用iBATIS删除记录。

我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

假设这个表是有两条记录如下：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

| 2 | Roma | Ali | 3000 |

+----+------------+-----------+--------+

2 row in set (0.00 sec)

**Employee POJO 类:**

要执行删除操作，就需要修改Employee.java文件。因此，让我们保持它不变，在上一章我们使用过。

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

/\* Here are the required method definitions \*/

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return first\_name;

}

public void setFirstName(String fname) {

this.first\_name = fname;

}

public String getLastName() {

return last\_name;

}

public void setlastName(String lname) {

this.last\_name = lname;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

} /\* End of Employee \*/

**Employee.xml 文件:**

要定义使用iBATIS SQL映射语句中，我们将增加<Delete>键标签Employee.xml文件，这个标签定义中，我们会定义将用于在IbatisDelete.java文件执行SQL DELETE查询数据库的“id”。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<insert id="insert" parameterClass="Employee">

INSERT INTO EMPLOYEE(first\_name, last\_name, salary)

values (#first\_name#, #last\_name#, #salary#)

<selectKey resultClass="int" keyProperty="id">

select last\_insert\_id() as id

</selectKey>

</insert>

<select id="getAll" resultClass="Employee">

SELECT \* FROM EMPLOYEE

</select>

<update id="update" parameterClass="Employee">

UPDATE EMPLOYEE

SET first\_name = #first\_name#

WHERE id = #id#

</update>

<delete id="delete" parameterClass="int">

DELETE FROM EMPLOYEE

WHERE id = #id#

</delete>

</sqlMap>

**IbatisDelete.java 文件:**

文件将应用程序级别的逻辑从Employee表中删除记录：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisDelete{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

/\* This would delete one record in Employee table. \*/

System.out.println("Going to delete record.....");

int id = 1;

smc.delete("Employee.delete", id );

System.out.println("Record deleted Successfully ");

System.out.println("Going to read records.....");

List <Employee> ems = (List<Employee>)

smc.queryForList("Employee.getAll", null);

Employee em = null;

for (Employee e : ems) {

System.out.print(" " + e.getId());

System.out.print(" " + e.getFirstName());

System.out.print(" " + e.getLastName());

System.out.print(" " + e.getSalary());

em = e;

System.out.println("");

}

System.out.println("Records Read Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述软件。请确保已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisDelete.java如上图所示，并编译它。
* 执行IbatisDelete二进制文件来运行程序。

得到以下结果，ID=1将在EMPLOYEE表中被删除，并读取EMPLOYEE表中的一条记录。

Going to delete record.....

Record deleted Successfully

Going to read records.....

2 Roma Ali 3000

Records Read Successfully

**iBATIS结果映射**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**6**

[上一篇](http://www.yiibai.com/ibatis/ibatis_delete_operation.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_stored_procedures.html)

resultMap的元素是在iBATIS的最重要和最强大的元素。您可以通过使用iBATIS的结果映射减少高达90％的JDBC编码，在某些情况下，可以让你做JDBC不支持的事情。

ResultMaps的设计是这样的简单语句不需要明确的结果映射，以及更复杂的报表要求不超过绝对必要说明的关系。

本章将只给你一个简单的介绍iBATIS的结果映射。

我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

此表有如下两条记录：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

| 2 | Roma | Ali | 3000 |

+----+------------+-----------+--------+

2 row in set (0.00 sec)

**Employee POJO 类:**

使用iBATIS的结果映射，需要修改Employee.java文件。因此，让我们保持它，因为它已经在前一章中使用过。

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

/\* Here are the required method definitions \*/

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return first\_name;

}

public void setFirstName(String fname) {

this.first\_name = fname;

}

public String getLastName() {

return last\_name;

}

public void setlastName(String lname) {

this.last\_name = lname;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

} /\* End of Employee \*/

**Employee.xml 文件:**

在这里，我们将修改Employee.xml文件介绍<resultMap></ resultMap>标记。这个标签就必须在我们<select>标记的resultMap属性运行此结果映射这是需要一个id。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<!-- Perform Insert Operation -->

<insert id="insert" parameterClass="Employee">

INSERT INTO EMPLOYEE(first\_name, last\_name, salary)

values (#first\_name#, #last\_name#, #salary#)

<selectKey resultClass="int" keyProperty="id">

select last\_insert\_id() as id

</selectKey>

</insert>

<!-- Perform Read Operation -->

<select id="getAll" resultClass="Employee">

SELECT \* FROM EMPLOYEE

</select>

<!-- Perform Update Operation -->

<update id="update" parameterClass="Employee">

UPDATE EMPLOYEE

SET first\_name = #first\_name#

WHERE id = #id#

</update>

<!-- Perform Delete Operation -->

<delete id="delete" parameterClass="int">

DELETE FROM EMPLOYEE

WHERE id = #id#

</delete>

<!-- Using ResultMap -->

<resultMap id="result" class="Employee">

<result property="id" column="id"/>

<result property="first\_name" column="first\_name"/>

<result property="last\_name" column="last\_name"/>

<result property="salary" column="salary"/>

</resultMap>

<select id="useResultMap" resultMap="result">

SELECT \* FROM EMPLOYEE

WHERE id=#id#

</select>

</sqlMap>

**IbatisResultMap.java 文件:**

文件将应用程序级别的逻辑，从使用结果映射Employee表中读取记录：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisResultMap{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

int id = 1;

System.out.println("Going to read record.....");

Employee e = (Employee)smc.queryForObject

("Employee.useResultMap", id);

System.out.println("ID: " + e.getId());

System.out.println("First Name: " + e.getFirstName());

System.out.println("Last Name: " + e.getLastName());

System.out.println("Salary: " + e.getSalary());

System.out.println("Record read Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述软件。请确保您已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisResultMap.java如上图所示，并编译它。
* 执行IbatisResultMap二进制文件来运行程序。

会得到下面的结果是对EMPLOYEE表的读操作。

Going to read record.....

ID: 1

First Name: Zara

Last Name: Ali

Salary: 5000

Record read Successfully

**iBATIS存储过程**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**0**

[上一篇](http://www.yiibai.com/ibatis/ibatis_result_maps.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_dynamic_sql.html)

使用iBATIS配置来调用存储过程。为了理解这一章，首先需要了解我们是如何在MySQL中创建一个存储过程。

在继续对本章学习之前，可以通过[MySQL存储过程](http://www.yiibai.com/jdbc/jdbc-stored-procedure.html)。

我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

让我们在MySQL数据库中创建以下存储过程。

DELIMITER $$

DROP PROCEDURE IF EXISTS `testdb`.`getEmp` $$

CREATE PROCEDURE `testdb`.`getEmp`

(IN empid INT)

BEGIN

SELECT \* FROM EMPLOYEE

WHERE ID = empid;

END $$

DELIMITER;

考虑EMPLOYEE表是有两条记录如下：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

| 2 | Roma | Ali | 3000 |

+----+------------+-----------+--------+

2 row in set (0.00 sec)

**Employee POJO 类:**

使用存储过程，你就需要修改Employee.java文件。因此，让我们保持它，因为它是在前一章。

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

/\* Here are the required method definitions \*/

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return first\_name;

}

public void setFirstName(String fname) {

this.first\_name = fname;

}

public String getLastName() {

return last\_name;

}

public void setlastName(String lname) {

this.last\_name = lname;

}

public int getSalary() {

return salary;

}

public void setSalary(int salary) {

this.salary = salary;

}

} /\* End of Employee \*/

**Employee.xml 类:**

在这里，我们将修改Employee.xml文件介绍<procedure></procedure>和<parameterMap></parameterMap>标记。这里<procedure></procedure>标签将有一个id，我们会用我们的应用程序来调用存储过程。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<!-- Perform Insert Operation -->

<insert id="insert" parameterClass="Employee">

INSERT INTO EMPLOYEE(first\_name, last\_name, salary)

values (#first\_name#, #last\_name#, #salary#)

<selectKey resultClass="int" keyProperty="id">

select last\_insert\_id() as id

</selectKey>

</insert>

<!-- Perform Read Operation -->

<select id="getAll" resultClass="Employee">

SELECT \* FROM EMPLOYEE

</select>

<!-- Perform Update Operation -->

<update id="update" parameterClass="Employee">

UPDATE EMPLOYEE

SET first\_name = #first\_name#

WHERE id = #id#

</update>

<!-- Perform Delete Operation -->

<delete id="delete" parameterClass="int">

DELETE FROM EMPLOYEE

WHERE id = #id#

</delete>

<!-- To call stored procedure. -->

<procedure id="getEmpInfo" resultClass="Employee"

parameterMap="getEmpInfoCall">

{ call getEmp( #acctID# ) }

</procedure>

<parameterMap id="getEmpInfoCall" class="map">

<parameter property="acctID" jdbcType="INT"

javaType="java.lang.Integer" mode="IN"/>

</parameterMap>

</sqlMap>

**IbatisSP.java 文件:**

文件将应用程序级别的逻辑读取使用结果映射Employee表员工的姓名name：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisSP{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

int id = 1;

System.out.println("Going to read employee name.....");

Employee e = (Employee)smc.queryForObject

("Employee.getEmpInfo", id);

System.out.println("First Name: " + e.getFirstName());

System.out.println("Record name Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述应用程序。请确保您在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisSP.java如上图所示，并编译它。
* 执行IbatisSP二进制文件来运行程序。

得到以下结果：

Going to read record.....

ID: 1

First Name: Zara

Last Name: Ali

Salary: 5000

Record read Successfully

**iBATIS动态SQL**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**8**

[上一篇](http://www.yiibai.com/ibatis/ibatis_stored_procedures.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_debugging.html)

使用动态查询是iBatis一个非常强大的功能。有时你已经改变WHERE子句条件的基础上你的参数对象的状态。在这种情况下的iBATIS提供了一组可以映射语句中使用，以提高SQL语句的重用性和灵活性的动态SQL标签。

所有的逻辑是使用一些额外的标签放在：XML文件。下面是一个例子，其中的SELECT语句将努力在两个方面：

* 如果想传递一个ID，然后它会返回所有与该ID的记录，
* 否则，将返回所有雇员ID为NULL的记录。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<select id="findByID" resultClass="Employee">

SELECT \* FROM EMPLOYEE

<dynamic prepend="WHERE ">

<isNull property="id">

id IS NULL

</isNull>

<isNotNull property="id">

id = #id#

</isNotNull>

</dynamic>

</select>

</sqlMap>

可以使用<isNotEmpty>标签如下检查条件。在此条件下将增加，只有当通过属性不为空。

..................

<select id="findByID" resultClass="Employee">

SELECT \* FROM EMPLOYEE

<dynamic prepend="WHERE ">

<isNotEmpty property="id">

id = #id#

</isNotEmpty>

</dynamic>

</select>

..................

如果想查询对id和/或雇员的名字选取。SELECT语句如下：

..................

<select id="findByID" resultClass="Employee">

SELECT \* FROM EMPLOYEE

<dynamic prepend="WHERE ">

<isNotEmpty prepend="AND" property="id">

id = #id#

</isNotEmpty>

<isNotEmpty prepend="OR" property="first\_name">

first\_name = #first\_name#

</isNotEmpty>

</dynamic>

</select>

..................

**例如：动态SQL**

下面的例子将展示如何编写SELECT语句中使用动态SQL。考虑，我们已经在MySQL下有EMPLOYEE表：

CREATE TABLE EMPLOYEE (

id INT NOT NULL auto\_increment,

first\_name VARCHAR(20) default NULL,

last\_name VARCHAR(20) default NULL,

salary INT default NULL,

PRIMARY KEY (id)

);

此表有如下只有一条记录：

mysql> select \* from EMPLOYEE;

+----+------------+-----------+--------+

| id | first\_name | last\_name | salary |

+----+------------+-----------+--------+

| 1 | Zara | Ali | 5000 |

| 2 | Roma | Ali | 3000 |

| 3 | Noha | Ali | 7000 |

+----+------------+-----------+--------+

3 row in set (0.00 sec)

**Employee POJO 类:**

要执行读取操作，让我们在Employee.java文件Employee类，如下所示：

public class Employee {

private int id;

private String first\_name;

private String last\_name;

private int salary;

/\* Define constructors for the Employee class. \*/

public Employee() {}

public Employee(String fname, String lname, int salary) {

this.first\_name = fname;

this.last\_name = lname;

this.salary = salary;

}

/\* Here are the method definitions \*/

public int getId() {

return id;

}

public String getFirstName() {

return first\_name;

}

public String getLastName() {

return last\_name;

}

public int getSalary() {

return salary;

}

} /\* End of Employee \*/

**Employee.xml 文件:**

要定义使用iBATIS SQL映射语句，我们将增加在以下文件Employee.xml修改<select>标记和这个标签定义，我们将定义一个“id”，这将被用于IbatisReadDy.java文件上执行动态SQL的SELECT查询数据库。

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE sqlMap

PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"

"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="Employee">

<select id="findByID" resultClass="Employee">

SELECT \* FROM EMPLOYEE

<dynamic prepend="WHERE ">

<isNotNull property="id">

id = #id#

</isNotNull>

</dynamic>

</select>

</sqlMap>

上面的SELECT语句将努力在两个方面（一）如果想传递一个ID，然后将相应的编号（二）返回的记录，否则将返回所有记录。

**IbatisReadDy.java 文件:**

文件将应用程序级别的逻辑从Employee表读出的条件记录：

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisReadDy{

public static void main(String[] args)

throws IOException,SQLException{

Reader rd=Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc=SqlMapClientBuilder.buildSqlMapClient(rd);

/\* This would read all records from the Employee table.\*/

System.out.println("Going to read records.....");

Employee rec = new Employee();

rec.setId(1);

List <Employee> ems = (List<Employee>)

smc.queryForList("Employee.findByID", rec);

Employee em = null;

for (Employee e : ems) {

System.out.print(" " + e.getId());

System.out.print(" " + e.getFirstName());

System.out.print(" " + e.getLastName());

System.out.print(" " + e.getSalary());

em = e;

System.out.println("");

}

System.out.println("Records Read Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述应用。请确保已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisReadDy.java如上图所示，并编译它。
* 执行IbatisReadDy二进制文件来运行程序。

会得到下面的结果，并且将记录从EMPLOYEE表中读取。

Going to read records.....

1 Zara Ali 5000

Record Reads Successfully

试试上面的例子中通过传递空值作为smc.queryForList（“Employee.findByID”，NULL）。

**iBATIS OGNL 表达式**

iBATIS的提供了强大的基于OGNL的表达式来消除其他元素。

* if 语句
* choose, when, otherwise 语句
* where 语句
* foreach 语句

**if语句：**

最常见的事情在动态SQL是有条件地包括一个where子句的一部分。例如：

<select id="findActiveBlogWithTitleLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

WHERE state = 'ACTIVE.

<if test="title != null">

AND title like #{title}

</if>

</select>

这条语句会提供功能的可选的文本搜索类型。如果没有传递title，那么所有激活的博客将被退回。但是，如果传递一个标题，它会寻找标题以like 给定的条件。

可以包括多个if条件如下：

最常见的事情在动态SQL是有条件地包括一个where子句的一部分。例如：

<select id="findActiveBlogWithTitleLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

WHERE state = 'ACTIVE.

<if test="title != null">

AND title like #{title}

</if>

<if test="author != null">

AND author like #{author}

</if>

</select>

**choose, when, otherwise 语句:**

iBATIS提供了一个选择的元素，它类似于Java的switch语句。这有助于选择很多种情况。

下面的例子将只搜索标题上如果提供，那么只有由作者如果已提供。如果没有提供，让我们只返回精选的博客：

<select id="findActiveBlogWithTitleLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

WHERE state = 'ACTIVE.

<choose>

<when test="title != null">

AND title like #{title}

</when>

<when test="author != null and author.name != null">

AND author like #{author}

</when>

<otherwise>

AND featured = 1

</otherwise>

</choose>

</select>

**where 语句:**

如果我们看一下前面的例子中，如果没有一个条件满足会发生什么事？最终SQL看起来像这样：

SELECT \* FROM BLOG

WHERE

这会失败，但iBATIS有一个简单的改变一个简单的解决方案，让一切工作正常：

<select id="findActiveBlogLike"

parameterType="Blog" resultType="Blog">

SELECT \* FROM BLOG

<where>

<if test="state != null">

state = #{state}

</if>

<if test="title != null">

AND title like #{title}

</if>

<if test="author != null>

AND author like #{author}

</if>

</where>

</select>

where元素知道只有插入WHERE ，如果有一个由含标签返回的任何内容。此外，如果该内容开头AND或OR，它知道剥离其关闭。

**foreach语句：**

foreach元素是非常强大的，并允许你指定一个集合，声明可以在元素的体内可用于项目和索引变量。

它也允许你指定打开和关闭的字符串，并添加一个分隔符放置在迭代之间。可以建立一个IN条件如下：

<select id="selectPostIn" resultType="domain.blog.Post">

SELECT \*

FROM POST P

WHERE ID in

<foreach item="item" index="index" collection="list"

open="(" separator="," close=")">

#{item}

</foreach>

</select>

**iBATIS调试**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**0**

[上一篇](http://www.yiibai.com/ibatis/ibatis_dynamic_sql.html)[下一篇](http://www.yiibai.com/ibatis/ibatis_vs_hibernate.html)

这是很容易，同时与iBATIS的工作程序进行调试。 iBATIS有内置的日志支持，并适用于下列日志库，并在这个顺序搜索他们。

* Jakarta Commons日志记录（JCL）。
* Log4J
* JDK 日志

可以使用任何上面列出的库在iBATIS。

**调试和Log4J：**

假设你要使用Log4J，这是最好用的日志记录。继续操作之前，需要交叉检查以下几点：

* Log4J JAR 文件 (log4j-{version}.jar) 应在CLASSPATH中。
* 必须在CLASSPATH中提供log4j.properties。

下面是一个log4j.properties文件。请注意，某些行被注释掉了。你可以取消他们，如果你需要额外的调试信息。

# Global logging configuration

log4j.rootLogger=ERROR, stdout

log4j.logger.com.ibatis=DEBUG

# shows SQL of prepared statements

#log4j.logger.java.sql.Connection=DEBUG

# shows parameters inserted into prepared statements

#log4j.logger.java.sql.PreparedStatement=DEBUG

# shows query results

#log4j.logger.java.sql.ResultSet=DEBUG

#log4j.logger.java.sql.Statement=DEBUG

# Console output

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n

可以找到完整的Log4J文档，从Apaches 网站： [Log4J 文档](http://logging.apache.org/log4j/).

**iBATIS 调试例子：**

下面的Java类是一个非常简单的例子，初始化，然后使用Java应用程序Log4J的日志库。它位于CLASSPATH中上面提到的属性文件。

import org.apache.log4j.Logger;

import com.ibatis.common.resources.Resources;

import com.ibatis.sqlmap.client.SqlMapClient;

import com.ibatis.sqlmap.client.SqlMapClientBuilder;

import java.io.\*;

import java.sql.SQLException;

import java.util.\*;

public class IbatisUpdate{

static Logger log = Logger.getLogger(

IbatisUpdate.class.getName());

public static void main(String[] args)

throws IOException,SQLException{

Reader rd = Resources.getResourceAsReader("SqlMapConfig.xml");

SqlMapClient smc = SqlMapClientBuilder.buildSqlMapClient(rd);

/\* This would insert one record in Employee table. \*/

log.info("Going to update record.....");

Employee rec = new Employee();

rec.setId(1);

rec.setFirstName( "Roma");

smc.update("Employee.update", rec );

log.info("Record updated Successfully ");

log.debug("Going to read records.....");

List <Employee> ems = (List<Employee>)

smc.queryForList("Employee.getAll", null);

Employee em = null;

for (Employee e : ems) {

System.out.print(" " + e.getId());

System.out.print(" " + e.getFirstName());

System.out.print(" " + e.getLastName());

System.out.print(" " + e.getSalary());

em = e;

System.out.println("");

}

log.debug("Records Read Successfully ");

}

}

**编译和运行：**

下面是步骤来编译并运行上述软件。请确保您已在进行的编译和执行之前，适当地设置PATH和CLASSPATH。

* 创建Employee.xml如上所示。
* 创建Employee.java如上图所示，并编译它。
* 创建IbatisUpdate.java如上图所示，并编译它。
* 创建log4j.properties文件，如上图所示。
* 执行IbatisUpdate二进制文件来运行程序。

会得到下面的结果，并记录在EMPLOYEE表进行更新, 然后相同的记录将被从EMPLOYEE表中读出。

DEBUG [main] - Created connection 28405330.

DEBUG [main] - Returned connection 28405330 to pool.

DEBUG [main] - Checked out connection 28405330 from pool.

DEBUG [main] - Returned connection 28405330 to pool.

1 Roma Ali 5000

2 Zara Ali 5000

3 Zara Ali 5000

**调试方法：**

在上面的例子中，我们只使用info()方法，但可以使用以下任何一种方法按你的需要：

public void trace(Object message);

public void debug(Object message);

public void info(Object message);

public void warn(Object message);

public void error(Object message);

public void fatal(Object message);

**iBATIS和Hibernate区别**

 来源：翻译整理    **注：**由于水平有限，欢迎指正。转载请务必注明出处。

**0**

[上一篇](http://www.yiibai.com/ibatis/ibatis_debugging.html)[下一篇](http://www.yiibai.com/ibatis/ibator_introduction.html)

iBatis和Hibernate之间有着较大的差异，但两者解决方案很好，因为他们有特定的领域。我个人建议使用iBATIS的，如果：

* 你想创建自己的SQL，并愿意维持他们.
* 你的环境是由关系数据模型驱动的。
* 你的项目工作有复杂架构的。

简单地要使用Hibernate，如果：

* 你的环境是由对象模型驱动的，并希望自动生成的SQL。

要计算的一些区别：

* iBATIS :
  + 简单
  + 更快的开发时间
  + 灵活
  + 封装尺寸更小
* Hibernate:
  + 为你生成SQL，这意味着你不用花时间在SQL上。
  + 提供了许多更先进的高速缓存
  + 高可扩展性

另一个区别是，iBATIS利用SQL语句可能是依赖数据库，使用Hibernate的HQL是相对独立于数据库，它是更容易改变数据库。

Hibernate映射的Java作为POJO对象，iBatis将ResultSet映射，从JDBC API给出POJO OBJETS的数据库表。

如果您使用存储过程，那么在Hibernate中可以做到这一点，但它在iBATIS比较有点困难。作为一种替代的解决方案iBATIS的映射结果集对象，所以没必要去关心表结构。这非常适用于存储过程，非常适用于报表应用程序等

最后，Hibernate和iBATIS的都是开源的对象关系映射（ORM）在同行业中可用的工具。使用这些工具的取决于你。Hibernate和iBatis两者也有来自Spring框架的良好支持，以便它不应该是一个问题，选择其中之一。