iBATIS Tutorial

iBATIS - Home
iBATIS - Overview
iBATIS - Environment
iBATIS - Create Operation
iBATIS - Read Operation
iBATIS - Update Operation
iBATIS - Delete Operation
iBATIS - Result Maps
iBATIS - Stored Procedures
iBATIS - Dynamic SQL
iBATIS - Debugging
iBATIS - Hibernate
iBATOR- Introduction

iBATIS Useful Resources

iBATIS - Quick Guide
iBATIS - Useful Resources

Selected Reading

Developer's Best Practices
Effective Resume Writing
Computer Glossary
Who is Who

iBATIS - Quick Guide

Previous Page Next Pa

iBATIS is a persistence framework which automates the mapping between SQL datab objects in Java, .NET, and Ruby on Rails. The mappings are decoupled from the applic by packaging the SQL statements in XML configuration files.

iBATIS Design Philosophies:

iBATIS comes with the following design philosophies:

Simplicity: iBATIS is widely regarded as being one of the simplest persistence fr available today.

Fast Development: iBATIS's philosophy is to do all it can to facilitate l development.

Portability: iBATIS can be implemented for nearly any language or platform like Ja and C# for Microsoft .NET.

Independent Interfaces: iBATIS provides database-independent interfaces and help the rest of the application remain independent of any persistence-related resour

Open source: iBATIS is free and an open source software.

Advantages of IBATIS

Here are few advantages of using IBATIS:

Suppports Stored procedures: iBATIS encapsulates SQL in the form of stored p so that business logic is kept out of the database, and the application is easier to d test, and is more portable.

Supports Inline SQL: No precompiler is needed, and you have full access to features of SQL.

Supports Dynamic SQL: iBATIS provides features for dynamically building SQ based on parameters.

Supports O/RM: iBATIS supports many of the same features as an O/RM tool, su loading, join fetching, caching, runtime code generation, and inheritance

iBATIS vs Hibernate:

There are major differences between iBatis and Hibernate but both the solutions work v their specific domain. Personally I would suggest you should use iBATIS if:

You want to create your own SQL's and are willing to maintain them.

your environment is driven by relational data model.

you have to work existing and complex schema's.

And simply use Hibernate if:

Your environment is driven by object model and wants generates SQL automatically.

To count there are few differences:

iBATIS is:

Simpler

Faster development time

Flixable

Much smaller in package size

Hibernate:

Generates SQL for you which means you don't spend time on SQL

Provides much more advance cache

Highly scalable

Other difference is that iBATIS makes use of SQL which could be database dependent where as Hibernate makes use of HQL which is relatively independent of databases and it is easier to change db in Hibernate.

Hibernate maps your Java POJO objects to the Database tables where as iBatis maps the ResultSet from JDBC API to your POJO Objets.

If you are using stored procedures, well you can do it in Hibernate but it is little difficult in comparision of iBATIS. As an alternative solution iBATIS maps results sets to objects, so no need to care about table structures. This works very well for stored procedures, works very well for reporting applications, etc

Finally, Hibernate and iBATIS both are open source Object Relational Mapping(ORM) tools available in the industry. Use of each of these tools depends on the context you are using them. Hibernate and iBatis both also have good support from SPRING framework so it should not be a problem to chose one of them.

iBATIS Installation:

Here are the simple steps you would need to carry out to install iBATIS on your Linux machine:

Download latest version of iBATIS from Download iBATIS.

Unzip the downloaded file to extract .jar file from the bundle and keep it in appropriate lib directory.

Set PATH and CLASSPATH variables at the extracted .jar file(s) appropriately.

Here are the steps I carried out on my linux machine after downloading iBATIS binary file:

```
$ 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
```

Create SqlMapConfig.xml

Consider the following:

We are going to use JDBC to access the database **testdb**.

JDBC driver for MySQL is "com.mysql.jdbc.Driver".

Connection URL is "jdbc:mysql://localhost:3306/testdb".

We would use username and password is "root" and "root".

Our sql statement mappings for all the operations would be described in "Employee.xml".

Based on the above assumption we have to create an XML configuration file with name **SqlMapConfig.xml** with the following content. This is where you need to provide all configurations required for iBatis:

It is important that both the files SqlMapConfig.xml and Employee.xml should be present in the class path. For now we would keep Employee.xml file empty and we would cover its content in subsequent chapters.

```
<?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>
```

iBATIS Practical Exmaple:

This example would show you how you can UPDATE and READ records from a MySQL database

table using iBATIS.

Consider we have following EMPLOYEE table in MySQL database:

```
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)
);
```

This table is having only one record as follows:

```
mysql> select * from EMPLOYEE;
+----+------------+-----------+--------+
| id | first_name | last_name | salary |
+----+------------+-----------+--------+
|  1 | Zara       | Ali       |   5000 |
+----+------------+-----------+--------+
1 row in set (0.00 sec)
```

Employee POJO Class:

Write the following Employee.java file:

```
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 File:

To define SQL mapping statement using iBATIS, we would add <update> and <select> tags in Employee.xml file and inside these tag definition we would define their IDs which will be used in IbatisUpdate.java file for executing SQL UPDATE and READ queries on database.

```
<?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="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 File:

This file would have application level logic to update and read records from the Employee table:

```
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 ");

  }
}
```

Compilation and Run:

Here are the steps to compile and run the above mentioned software. Make sure you have set PATH and CLASSPATH appropriately before proceeding for the compilation and execution.

Create SqlMapConfig.xml as shown above.

Create Employee.xml as shown above.

Create Employee.java as shown above and compile it.

Create IbatisUpdate.java as shown above and compile it.

Execute IbatisUpdate binary to run the program.

You would get following result, and a record would be updated in EMPLOYEE table and later same record would be read from the EMPLOYEE table.

```
Going to update record.....
Record updated Successfully
Going to read records.....
  1  Roma  Ali  5000
Records Read Successfully
```

Previous Page | Print Version | PDF Version | Next Page