iBATIS

database application framework

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

iBATIS is a persistence framework which automates the mapping between SQL databases and objects in Java, .NET, and Ruby on Rails. iBATIS makes it easier to build better database-oriented applications more quickly and with less code.

Audience

This tutorial is designed for Java programmers who would like to understand the iBATIS framework in detail along with its architecture and actual usage.

Prerequisites

Before proceeding with this tutorial, you should have a good understanding of Java programming language. As you are going to deal with SQL mapping, it is required that you have adequate exposure to SQL and Database concepts.

Copyright & Disclaimer

Table of Contents

1. OVERVIEW

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

iBATIS is a lightweight framework and persistence API good for persisting POJOs(Plain Old Java Objects).

iBATIS is what is known as a data mapper and takes care of mapping the parameters and results between the class properties and the columns of the database table.

A significant difference between iBATIS and other persistence frameworks such as Hibernate is that iBATIS emphasizes the use of SQL, while other frameworks typically use a custom query language such has the Hibernate Query Language (HQL) or Enterprise JavaBeans Query Language (EJB QL).

iBATIS Design Philosophies

iBATIS comes with the following design philosophies:

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

- **Fast Development:** iBATIS does all it can to facilitate hyper-fast development.

- **Portability:** iBATIS can be implemented for nearly any language or platform such as Java, Ruby, and C# for Microsoft .NET.

- **Independent Interfaces:** iBATIS provides database-independent interfaces and APIs that help the rest of the application remain independent of any persistence-related resources.

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

Advantages of iBATIS

iBATIS offers the following advantages:

- **Supports stored procedures:** iBATIS encapsulates SQL in the form of stored procedures so that business logic is kept out of the database, and the application is easier to deploy and test, and is more portable.

- **Supports inline SQL:** No precompiler is needed, and you have full access to all of the features of SQL.

- **Supports dynamic SQL:** iBATIS provides features for dynamically building SQL queries based on parameters.

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

iBATIS makes use of JAVA programming language while developing database oriented application. Before proceeding further, make sure that you understand the basics of procedural and object-oriented programming: control structures, data structures and variables, classes, objects, etc.

To understand JAVA in detail you can go through our JAVA Tutorial.

2. ENVIRONMENT

You would have to set up a proper environment for iBATIS before starting off with actual development work. This chapter explains how to set up a working environment for iBATIS.

iBATIS Installation

Carry out the following simple steps to install iBATIS on your Linux machine:

- Download the latest version of iBATIS from **Download iBATIS**.
- Unzip the downloaded file to extract .jar file from the bundle and keep them in appropriate lib directory.
- Set PATH and CLASSPATH variables at the extracted .jar file(s) appropriately.

```
$ 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
```

Database Setup

Create an EMPLOYEE table in any MySQL database using the following syntax:

```
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)

        );
```

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 as "root" and "root" respectively.
- Our SQL statement mappings for all the operations would be described in "Employee.xml".

Based on the above assumptions, 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 contents 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>
```

You can set the following optional properties as well using SqlMapConfig.xml file:

```
<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"/>
```

3. CREATE OPERATION

To perform any Create, Write, Update, and Delete (CRUD) operation using iBATIS, you would need to create a Plain Old Java Objects (POJO) class corresponding to the table. This class describes the objects that will "model" database table rows.

The POJO class would have implementation for all the methods required to perform desired operations.

Let us assume we have the following EMPLOYEE table in 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)
);
```

Employee POJO Class

We would create an Employee class in Employee.java file as follows:

```
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 */
```

You can define methods to set individual fields in the table. The next chapter explains how to get the values of individual fields.

Employee.xml File

To define SQL mapping statement using iBATIS, we would use <insert> tag and inside this tag definition, we would define an "id" which will be used in IbatisInsert.java file for executing SQL INSERT query 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">
<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>
```

Here **parameterClass:** could take a value as *string, int, float, double,* or any *class object* based on requirement. In this example, we would pass Employee object as a parameter while calling *insert* method of SqlMap class.

If your database table uses an IDENTITY, AUTO_INCREMENT, or SERIAL column or you have defined a SEQUENCE/GENERATOR, you can use the <selectKey> element in an <insert> statement to use or return that database-generated value.

IbatisInsert.java File

This file would have application level logic to insert records in 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 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 ");

  }
}
```

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 compilation and execution.

- Create Employee.xml as shown above.
- Create Employee.java as shown above and compile it.
- Create IbatisInsert.java as shown above and compile it.
- Execute IbatisInsert binary to run the program.

You would get the following result, and a record would be created in the EMPLOYEE table.

```
$java IbatisInsert
Going to insert record.....
Record Inserted Successfully
```

If you check the EMPLOYEE table, it should display the following result:

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

4. READ OPERATION

We discussed, in the last chapter, how to perform CREATE operation on a table using iBATIS. This chapter explains how to read a table using iBATIS.

We have the following EMPLOYEE table in 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)
);
```

This table has 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

To perform read operation, we would modify the Employee class in Employee.java as follows:

```
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 File

To define SQL mapping statement using iBATIS, we would add <select> tag in Employee.xml and inside this tag definition, we would define an "id" which will be used in IbatisRead.java file for executing SQL SELECT query on database.