IBATOR - INTRODUCTION

iBATOR is a code generator for iBATIS. iBATOR introspects one or more database tables and generates iBATIS artifacts that can be used to access the tables.

Later you can write your custom SQL code or stored procedure to meet your requirements. iBATOR generates the following artifacts −

- SqlMap XML Files
- Java Classes to match the primary key and fields of the table*s*
- DAO Classes that use the above objects *optional*

iBATOR can run as a standalone JAR file, or as an Ant task, or as an Eclipse plugin. This tutorial describes the simplest way of generating iBATIS configuration files from command line.

Download iBATOR

Download the standalone JAR if you are using an IDE other than Eclipse. The standalone JAR includes an Ant task to run iBATOR, or you can run iBATOR from the command line of Java code.

- You can download zip file from Download iBATOR.
- You can check online documentation − iBATOR Documentation.

Generating Configuration File

To run iBATOR, follow these steps −

Step 1

Create and fill a configuration file ibatorConfig.xml appropriately. At a minimum, you must specify −

- A **<jdbcConnection>** element to specify how to connect to the target database.
- A **<javaModelGenerator>** element to specify the target package and the target project for the generated Java model objects.
- A **<sqlMapGenerator>** element to specify the target package and the target project for the generated SQL map files.
- A **<daoGenerator>** element to specify the target package and the target project for the generated DAO interfaces and classes
  youmayomitthe < daoGenerator > elementifyoudon't wishtogenerateDAOs.
- At least one database **<table>** element

NOTE − See the XML Configuration File Reference page for an example of an iBATOR configuration file.

Step 2

Save the file in a convenient location, for example, at: \temp\ibatorConfig.xml.

Step 3

Now run iBATOR from the command line as follows −

```
java -jar abator.jar -configfile \temp\abatorConfig.xml -overwrite
```

It will tell iBATOR to run using your configuration file. It will also tell iBATOR to overwrite any existing Java files with the same name. If you want to save any existing Java files, then omit the **−overwrite** parameter.

If there is a conflict, iBATOR saves the newly generated file with a unique name.

After running iBATOR, you need to create or modify the standard iBATIS configuration files to make use of your newly generated code. This is explained in the next section.

Tasks After Running iBATOR

After you run iBATOR, you need to create or modify other iBATIS configuration artifacts. The main tasks are as follows −

- Create or modify the SqlMapConfig.xml file.
- Create or modify the dao.xml file *onlyifyouareusingtheiBATISDAOFramework*.

Each task is described in detail below −

Updating the SqlMapConfig.xml File

iBATIS uses an XML file, commonly named SqlMapConfig.xml, to specify information for a database connection, a transaction management scheme, and SQL map XML files that are used in an iBATIS session.

iBATOR cannot create this file for you because it knows nothing about your execution environment. However, some of the items in this file relate directly to iBATOR generated items.

iBATOR specific needs in the configuration file are as follows −

- Statement namespaces must be enabled.
- iBATOR generated SQL Map XML files must be listed.

For example, suppose iBATOR has generated an SQL Map XML file called MyTable_SqlMap.xml, and that the file has been placed in the test.xml package of your project. The SqlMapConfig.xml file should have these entries −

```
<?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>
   <!-- Statement namespaces are required for Abator -->
   <settings useStatementNamespaces="true" />

   <!-- Setup the transaction manager and data source that are
   appropriate for your environment
   -->

   <transactionManager type="...">
      <dataSource type="...">
      </dataSource>
   </transactionManager>

   <!-- SQL Map XML files should be listed here -->
   <sqlMap resource="test/xml/MyTable_SqlMap.xml" />

</sqlMapConfig>
```

If there is more than one SQL Map XML file *asisquitecommon*, then the files can be listed in any order with repeated <sqlMap> elements after the <transactionManager> element.

Updating the dao.xml File

The iBATIS DAO framework is configured by an xml file commonly called dao.xml.

The iBATIS DAO framework uses this file to control the database connection information for DAOs, and also to list the DAO implementation classes and DAO interfaces.

In this file, you should specify the path to your SqlMapConfig.xml file, and all the iBATOR generated DAO interfaces and implementation classes.

For example, suppose iBATOR has generated a DAO interface called MyTableDAO and an implementation class called MyTableDAOImpl, and that the files have been placed in the test.dao package of your project.

The dao.xml file should have these entries −

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE daoConfig PUBLIC "-//ibatis.apache.org//DTD DAO Configuration 2.0//EN"
"http://ibatis.apache.org/dtd/dao-2.dtd">

<daoConfig>

   <context>

      <transactionManager type="SQLMAP">
         <property name="SqlMapConfigResource" value="test/SqlMapConfig.xml"/>
      </transactionManager>

      <!-- DAO interfaces and implementations should be listed here -->
      <dao interface="test.dao.MyTableDAO" implementation="test.dao.MyTableDAOImpl" />
   </context>

</daoConfig>
```

NOTE − This step is required only if you generated DAOs for the iBATIS DAO framework.