**Building a Database Test Plan**

In this section, you will learn how to create a basic [Test Plan](http://jmeter.apache.org/usermanual/build-test-plan.html) to test a database

Server. You will create fifty users that send 2 SQL requests to the database server. Also, you will tell the users to run their tests 100 times. So, the total number of requests is (50 users) x (2 requests) x (repeat 100 times) = 10'000 JDBC requests. To construct the Test Plan, you will use the following elements: [Thread Group](http://jmeter.apache.org/usermanual/test_plan.html#thread_group), [JDBC Request](http://jmeter.apache.org/usermanual/component_reference.html#JDBC_Request), [and Summary Report](http://jmeter.apache.org/usermanual/component_reference.html#Summary_Report).

This example uses the MySQl database driver. To use this driver, its containing .jar file (ex. mysql-connector-java-x.x.x-bin.jar) must be copied to the jmeter ./lib directory (see jmeter's

Classpath for more details).

**Adding Users**

* The first step you want to do with every JMeter Test Plan is to add a [Thread](http://jmeter.apache.org/usermanual/test_plan.html" \l "thread_group)

[Group](http://jmeter.apache.org/usermanual/test_plan.html" \l "thread_group) element. The Thread Group tells JMeter the number of users you want to simulate, how often the users should send requests, and the how many requests they should send.

* Go ahead and add the Thread Group element by first selecting the Test Plan, clicking your right mouse button to get the Add menu, and then select **Add** -> **ThreadGroup**.
* You should now see the Thread Group element under Test Plan. If you do not see the element, then "expand" the Test Plan tree by clicking on the Test Plan element.
* Next, you need to modify the default properties. Select the Thread Group element in the tree, if you have not already selected it. You should now see the Thread Group Control Panel in the right section of the JMeter window (see Figure 7.1 below)

[
Figure 7.1. Thread Group with Default Values](http://jmeter.apache.org/images/screenshots/jdbctest/threadgroup1.png)Figure 7.1. Thread Group with Default Values

* Start by providing a more descriptive name for our Thread Group. In the name field, enter JDBC Users.
* You will need a valid database, database table and user-level access to that table. In the example shown here, the database is **'cloud'** and the table name is **'VM\_instance'**.
* Next, increase the number of users to 50.
* In the next field, the Ramp-Up Period, leave the the value of 10 seconds. This property tells JMeter how long to delay between starting each user. For example, if you enter a Ramp-Up Period of 10 seconds, JMeter will finish starting all of your users by the end of the 10 seconds. So, if we have 50 users and a 10 second Ramp-Up Period, then the delay between starting users would be 200 milliseconds (10 seconds / 50 users = 0.2 user per second). If you set the value to 0, then JMeter will immediately start all of your users.
* Finally, enter a value of 100 in the Loop Count field. This property tells JMeter how many times to repeat your test. To have JMeter repeatedly run your Test Plan, select the Forever checkbox.
* In most application, you have to manually accept changes you make in a control panel. However, in jmeer the control panel automatically accepts your changes as you make them. If you change the name of an element, ther tree will be updated with the new text after you leave the control panel (for example, when selecting another tree element).

See Figure 7.2 for the completed JDBC Users Thread Group.

[
Figure 7.2. JDBC Users Thread Group](http://jmeter.apache.org/images/screenshots/jdbctest/threadgroup2.png)Figure 7.2. JDBC Users Thread Group

**Adding JDBC Requests**

Now that we have defined our users, it is time to define the tasks that they will be performing. In this section, you will specify the JDBC requests to perform.

Begin by selecting the JDBC Users element. Click your right mouse button to get the **Add** menu, and then select **Add** --> **Config Element** --> **JDBC Connection Configuration**. Then, select this new element to view its Control Panel (see Figure 7.3).

**Set up the following fields (these assume we will be using a MySQL database called 'cloud'):**

* Variable name (here: my Database) bound to pool. This needs to uniquely identify the configuration. It is used by the JDBC Sampler to identify the configuration to be used.
* Database URL: jdbc:mysql://ipOfTheServer:3306/cloud
* JDBC Driver class: com.mysql.jdbc.Driver
* Username: the username of database
* Password: password for the username

The other fields on the screen can be left as the defaults.

JMeter creates a database connection pool with the configuration settings as specified in the Control Panel. The pool is referred to in JDBC Requests in the 'Variable Name' field. Several different JDBC Configuration elements can be used, but they must have unique names. Every JDBC Request must refer to a JDBC Configuration pool. More than one JDBC Request can refer to the same pool.

[
Figure 7.3. JDBC Configuration](http://jmeter.apache.org/images/screenshots/jdbctest/jdbc-config.png)Figure 7.3. JDBC Configuration

Selecting the JDBC Users element again. Click your right mouse button to get the **Add** menu, and then select **Add** --> **Sampler** --> **JDBC Request**. Then, select this new element to view its Control Panel (see Figure 7.4).

[
Figure 7.4. JDBC Request](http://jmeter.apache.org/images/screenshots/jdbctest/JDBCRequest.png)Figure 7.4. JDBC Request

In our Test Plan, we will make two JDBC requests. The first one is for select all 'Running' VM instances, and the second is to select 'Expunging' VM instance (obviously you should change these two examples appropriate for your particular database). These are illustrated below.

Jmeter sends requests in the order that you add them to the tree.

**Start by editing the following properties (see Figure 7.5):**

* Change the Name to **'VM Running'**.
* Enter the Pool Name: **'my Database'** (same as in the configuration element)
* Enter the SQL Query String field.
* Enter the Parameter values field with **'Running'** value.
* Enter the Parameter types with **'VARCHAR'**.

[
Figure 7.5. JDBC Request for the first SQL request](http://jmeter.apache.org/images/screenshots/jdbctest/JDBCRequest2.png)Figure 7.5. JDBC Request for the first SQL request

**Next, add the second JDBC Request and edit the following properties (see Figure 7.6):**

* Change the Name to **'VM Expunging'**.
* Change the value of Parameter values to **'Expunging'**.

[
Figure 7.6. JDBC Request for the second request](http://jmeter.apache.org/images/screenshots/jdbctest/JDBCRequest3.png)

Figure 7.6. JDBC Request for the second request

**Adding a Listener to View/Store the Test Results**

* The final element you need to add to your Test Plan is a [Listener](http://jmeter.apache.org/usermanual/component_reference.html#listeners). This element is responsible for storing all of the results of your JDBC requests in a file and presenting the results.
* Select the **JDBC****Users**element and add a [Summary Report](http://jmeter.apache.org/usermanual/component_reference.html#Summary_Report) listener (**Add** --> **Listener** --> **Summary Report**).
* Save the test plan, and run the test with the menu **Run** --> **Start** or **Ctrl+R**

The listener shows the results.

[
Figure 7.7. Graph results Listener](http://jmeter.apache.org/images/screenshots/jdbctest/jdbc-results.png)Figure 7.7. Graph results Listener