14 Testing a database

14.1 About

As part of performance testing, we want to make sure we are testing all the components of our SUT at some point. When we test a website, it is likely that all the components are being used at some point, but we may also want to directly test individual parts such as web services or databases, so we can see the effect of any performance tuning that is done by the database administrator (DBA).

In this activity, you will do the following:

- Examine the database structure
- Create a new JMeter test
- Add a database connection to the test
- Add a JDBC request sampler
- Add listeners
- Run the test and examine the results

14.2 JDBC

JMeter can directly query databases using **JDBC** (Java Database Connectivity). JDBC is Java’s method of querying a database from within a Java application. It is a standard component of Java. Using JDBC means that the application doesn’t need to know anything about the database except how to connect to it.

Applications use JDBC by first creating a connection to the database. The connection specifies the server name, port number, database name and other information such as authentication credentials. If you don’t know this information, you will need to liaise with the DBA.

Once the connection is established, the application can send SQL queries directly to the database. Any valid SQL query can be sent (for example select, insert, update, delete queries) as long as the user has the appropriate permissions. Clearly some knowledge of the database schema (structure of database tables and columns) is required to do this- again the DBA may need to be involved.

In JMeter, the steps for setting up a database test are:

- Add the **JDBC Connection Configuration** element to a thread group to control the connection to the database
- Add **JDBC Request** samplers to query the database
- Add a **View Results Tree** listener to review the results of database queries
- Add a **Summary Report** listener to show performance statistics

14.3 Finding JDBC drivers

To access databases via JDBC, the correct JDBC drivers for the database being used must be installed beforehand. JDBC drivers exist for all popular databases, and can usually downloaded from the database vendor’s website. For example:

- MySQL : https://www.mysql.com/products/connector/ - when you get to the point of choosing an operating system, choose **Platform Independent**- this allows you to directly download the .jar file.
- Oracle : http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
- Microsoft SQL Server : https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server
- Microsoft Access : Microsoft do not produce an official JDBC driver for Access, but there are a couple of third-party drivers, including at least one open source driver available from http://ucanaccess.sourceforge.net/site.html

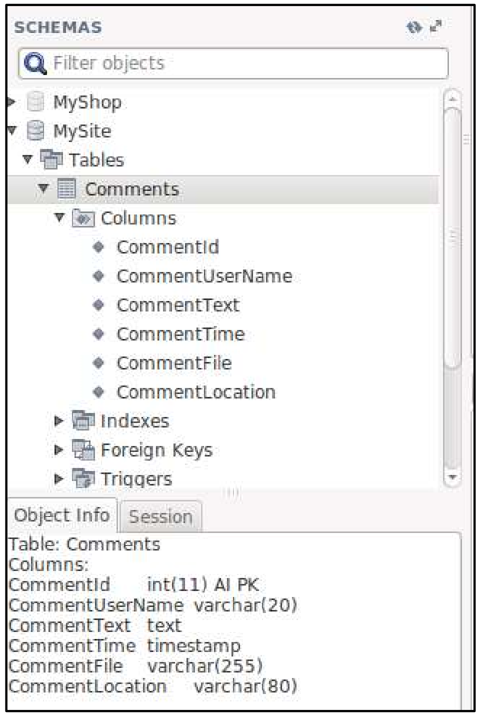
The driver must be placed in the JMeter **/lib** folder. In the training environment, the MySQL driver is already installed.

14.4 Examine the database structure

We need to know a bit about the database structure so we can send queries to the database. In the training environment, our database is based on MySQL.

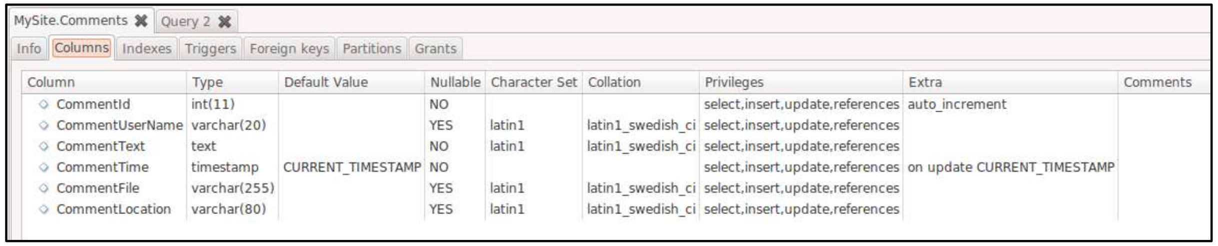
Open MySQL Workbench from the quick launch toolbar and connect to the database (quick reminder: **Database>Connect to Database…**, hostname **localhost**, port **3306**, username **root**, password **root**)

Browse the **Schemas** panel and find the **MySite** schema (the schema for the sample website) and browse to find the **Comments** table. The **Object Info** panel underneath shows brief information about the table.



```
Table: Comments
Columns:
CommentId      int(11) AI PK
CommentUserName varchar(20)
CommentText  text
CommentTime  timestamp
CommentFile    varchar(255)
CommentLocation    varchar(80)
```

Right click the table name and select **Table Inspector**. Full information about the table appears in the centre of the screen. Switch to the **Columns** tab to see information about the columns of the table.



Column	Type	Default Value	Nullable	Character Set	Collation	Privileges	Extra	Comments
CommentId	int(11)		NO			select,insert,update,references	auto_increment	
CommentUserName	varchar(20)		YES	latin1	latin1_swedish_ci	select,insert,update,references		
CommentText	text		NO	latin1	latin1_swedish_ci	select,insert,update,references		
CommentTime	timestamp	CURRENT_TIMESTAMP	NO			select,insert,update,references	on update CURRENT_TIMESTAMP	
CommentFile	varchar(255)		YES	latin1	latin1_swedish_ci	select,insert,update,references		
CommentLocation	varchar(80)		YES	latin1	latin1_swedish_ci	select,insert,update,references		

Close the MySQL Workbench when you are finished.

14.5 Create a new JMeter test

Open JMeter if you don’t have it open already.

Close any tests that are open.

14.6 Add elements to the test

14.6.1 Add a thread group

JDBC requests, like HTTP request samplers, are added to a thread group.

Add a thread group to the test- right click the Test Plan and choose **Add>Threads (Users)>Thread Group**

Set the properties of the thread group as follows:

Number of threads (users) : **10**

Ramp-Up Period (in seconds) : **20**

Loop Count : **20**

14.6.2 Add a JDBC connection

Right click the thread group and choose **Add>Config Element>JDBC Connection Configuration**

Fill in the connection information as follows:

Variable name for created pool : **CommentsDB** (this is the label that identifies the connection in JDBC samplers)

Database URL : **jdbc:mysql://localhost:3306/MySite**

JDBC Driver class : **com.mysql.jdbc.Driver**

Username : **root**

Password : **root**

14.6.3 Add a JDBC request sampler

JDBC request samplers send queries to the database.

Add a JDBC request sampler- right click the thread group and choose **Add>Sampler>JDBC Request**

Set the properties of the sampler as follows:

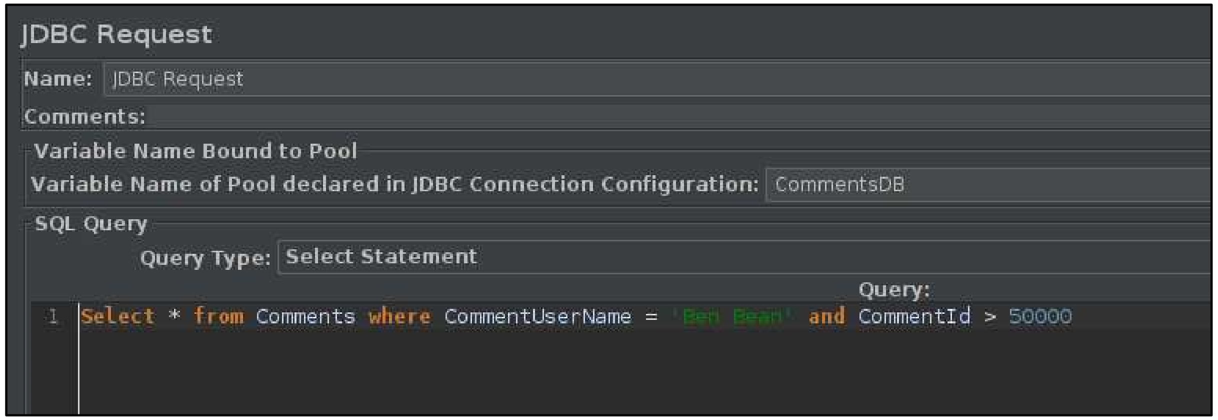
Variable name : **CommentsDB** (this matches the variable name specified in the connection configuration in the previous section)

Query Type : **Select Statement** (this is the default- examine the other entries in the list as well)

In the **Query** panel enter a valid SQL select statement for the Comments table- use the following statement but adjust the name and id numbers where necessary:

```
select * from Comments where CommentUserName = 'Ben Bean' and CommentId > 50000
```

Note that normally an SQL statement is ended with a semicolon (**;**) but in this sampler it is not necessary (and in fact the JMeter documentation specifically says it should not be added).


JDBC Request

Name: JDBC Request

Comments:

Variable Name Bound to Pool

Variable Name of Pool declared in JDBC Connection Configuration: CommentsDB

SQL Query

Query Type: Select Statement

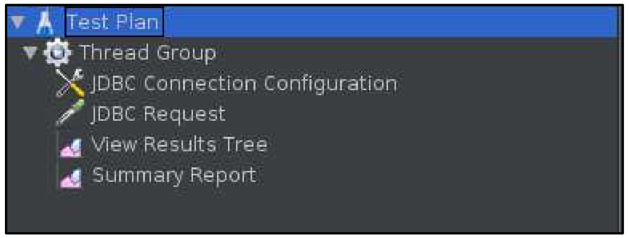
Query:

```
Select * from Comments where CommentUserName = 'Ben Bean' and CommentId > 50000
```

14.6.4 Add listeners
Listeners show the results of our samplers. We will add two listeners to show us the results of the JDBC requests and to show a summary of the test.

Add a View Results Tree listener- right click the thread group and choose **Add>Listener>View Results Tree**
Add a Summary Report listener- right click the thread group and choose **Add>Listener>Summary Report**
Leave the settings for the listeners as their defaults.

14.6.5 Checkpoint
At this point the Test Plan should look similar to this:

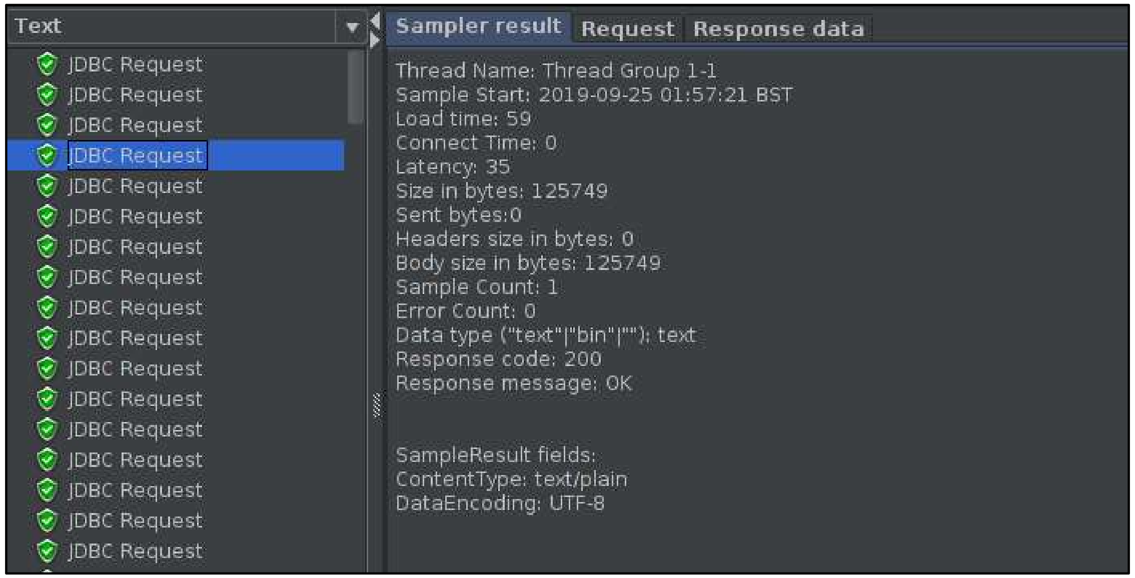


14.7 Save and run the test and examine the results

Save the test before continuing. Save it to the workspace folder with the name **database.jmx**.

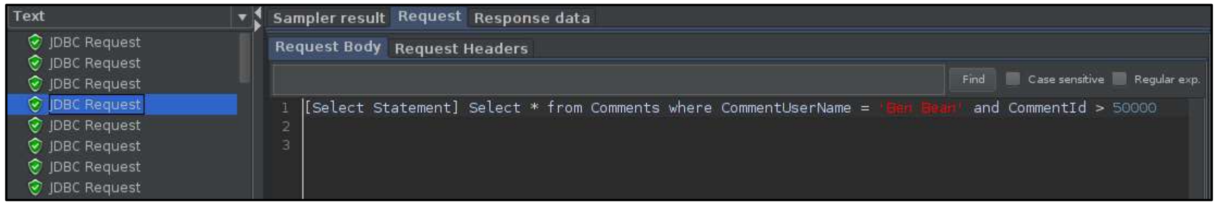
Run the test. As the test runs, examine the listeners.

In the View Results Tree listener, select a request and check the **Sampler result**, **Request** and **Response data** tabs.

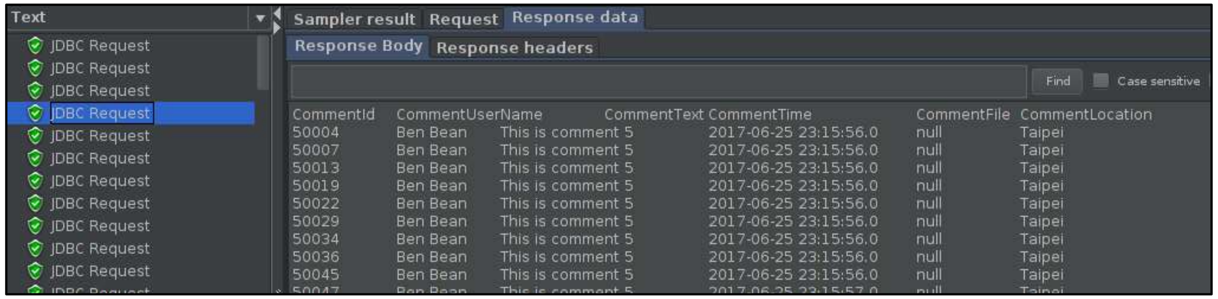


```
Thread Name: Thread Group 1-1
Sample Start: 2019-09-25 01:57:21 BST
Load time: 59
Connect Time: 0
Latency: 35
Size in bytes: 125749
Sent bytes:0
Headers size in bytes: 0
Body size in bytes: 125749
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"|""): text
Response code: 200
Response message: OK


SampleResult fields:
ContentType: text/plain
DataEncoding: UTF-8
```

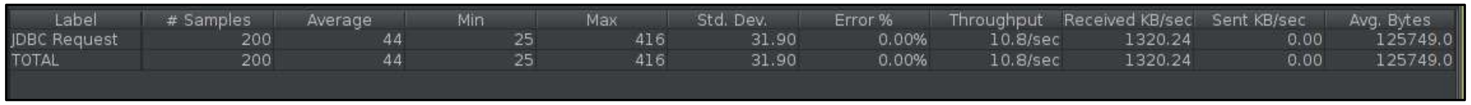


```
[Select Statement] Select * from Comments where CommentUserName = 'Ben Bean' and CommentId > 50000
```



```
CommentId  CommentUserName        CommentText CommentTime           CommentFile  CommentLocation
50004      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50007      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50013      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50019      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50022      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50029      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50034      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50036      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
50045      Ben Bean    This is comment 5      2017-06-25 23:15:56.0  null         Taipei
```

Open the Summary Report listener and examine the statistics.



```
Label         # Samples  Average  Min  Max  Std. Dev.  Error %  Throughput  Received KB/sec  Sent KB/sec  Avg. Bytes
JDBC Request        200       44   25  416      31.90    0.00%   10.8/sec          1320.24         0.00    125749.0
TOTAL               200       44   25  416      31.90    0.00%   10.8/sec          1320.24         0.00    125749.0
```

14.8 Explore more

14.8.1 Query the database performance

Most databases can provide performance data by sending commands directly to the database. For example, open the JDBC request sampler properties and change the query to show global status and run the test again. Examine the responses in the View Results Tree listener to see the data returned. Individual values can be queried. Change the query to show global status like ‘Queries’

Additionally MySQL has a built in performance monitoring system which stores information into a database called **performance_schema**. This database must be enabled on the server. Once it is enabled, it can be queried directly- for example change the JDBC request query to:

```
Select variable_value from performance_schema.global_status where variable_name = 'Queries'
```

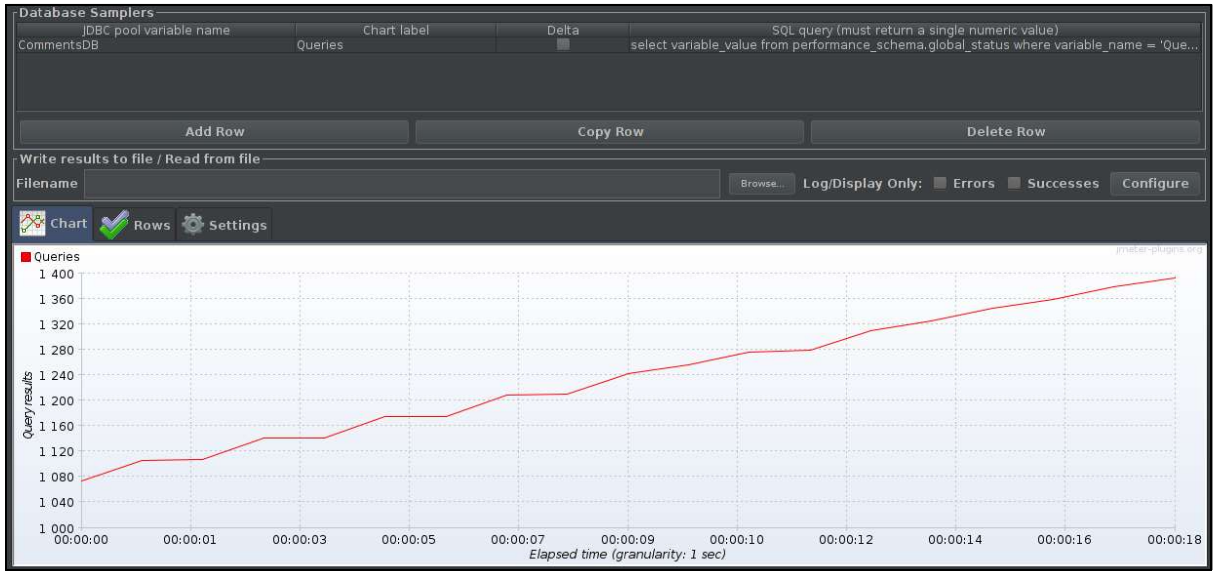
14.8.2 Use the DBMon plugin

The JMeter plugins set includes a plugin called **DBMon** which can plot a graph of database server performance. The values on the graph are based on the results of queries such as the one shown above (the query must return a single value).

Enable the plugin on the Plugins Manager (**Options>Plugins Manager**) and add the **DBMon Samples Collector** listener to the test.

Open the listener properties and add a row to the **Database Samplers** table. Use the pool name used previously (**CommentsDB**) and set the chart label to **Queries**. Use the query shown above.

Save and run the test and observe what is shown on the graph.



Database Samplers

JDBC pool variable name	Chart label	Delta	SQL query (must return a single numeric value)
CommentsDB	Queries	☐	select variable_value from performance_schema.global_status where variable_name = 'Que...

Investigate the performance_schema tables using the MySQL Workbench to see what other values are available. Refer to the MySQL documentation at https://dev.mysql.com/doc/refman/5.7/en/performance-schema.html for complete information.