

Module 8 - Java

Working with Databases

Advanced Java Certification Training

Akram M'Tir

1. 1. Write a programme to get the student's name for a given student's ID from the test.student table.
 - a. Use stored procedure (script provided in scripts/get_name.sql).

Database, Table

```
MariaDB [test]> describe STUDENT;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| ID    | int(11)       | NO   | PRI | NULL    | auto_increment |
| NAME  | varchar(255)  | YES  |     | NULL    |                |
| MARKS | int(11)       | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

MariaDB [test]> select * from STUDENT;
+-----+-----+-----+
| ID | NAME  | MARKS |
+-----+-----+-----+
| 1  | Tom   | 95    |
| 2  | John  | 75    |
| 3  | Alice | 90    |
| 4  | Bob   | 98    |
+-----+-----+-----+
```

Procedure Creation and query

```
MariaDB [test]> DROP PROCEDURE IF EXISTS get_name;
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> delimiter //
MariaDB [test]> CREATE PROCEDURE `get_name` (IN id INT, OUT name VARCHAR(20))
-> BEGIN
->     SELECT STUDENT.NAME INTO name FROM STUDENT WHERE STUDENT.ID=id;
-> END//
Query OK, 0 rows affected (0.00 sec)

MariaDB [test]> delimiter ;
MariaDB [test]>
MariaDB [test]> call get_name(3 , @n);
Query OK, 1 row affected (0.00 sec)

MariaDB [test]>
MariaDB [test]>
MariaDB [test]> select @n;
+-----+
| @n    |
+-----+
| Alice |
+-----+
1 row in set (0.00 sec)
```

Java Code

```
OutParamDemo.java TestConnectionMySQL.java dbconn.properties log4j.properties Crea
1 package module8.proc;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.IOException;
6 import java.sql.CallableStatement;
7 import java.sql.Connection;
8 import java.sql.DriverManager;
9 import java.sql.SQLException;
10 import java.util.Properties;
11 import org.apache.log4j.Logger;
12
13 public class OutParamDemo {
14
15     private static final Logger log4j = Logger.getLogger(OutParamDemo.class.getName());
16     private static final String DB_CONN_FILE = "resources/dbconn.properties";
17
18     public static void main(String[] args) {
19         Properties p = new Properties();
20
21         try {
22             p.load(new FileInputStream(new File(DB_CONN_FILE)));
23         } catch (IOException e) {
24             log4j.error("Error reading " + DB_CONN_FILE, e);
25             System.exit(-1);
26         }
27
28         log4j.debug("Connecting to database.");
29
30         try (Connection conn = DriverManager.getConnection(p.getProperty("url"), p);) {
31             log4j.debug("Connecting succeeded.");
32
33             String sqlStat = "CALL get_name(?, ?)";
34             int ID = 2;
35             try (CallableStatement proc = conn.prepareCall(sqlStat);) {
36                 proc.setInt(1, ID);
37                 proc.execute();
38
39                 String str = null;
40                 str = proc.getString(2);
41
42                 log4j.debug("ID : " + ID + " --> Name " + str + ".");
43             }
44         } catch (SQLException e) {
45             log4j.error("Query failed.", e);
46         } finally {
47             log4j.debug("Connection closed.");
48         }
49     }
50 }
51
52
```

Console Log4j output

```
Problems Console Javadoc Declaration Terminal Debug
<terminated> OutParamDemo [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/j
2018-05-25 02:56:17 main DEBUG module8.proc.OutParamDemo:34 - Connecting to database.
2018-05-25 02:56:18 main DEBUG module8.proc.OutParamDemo:37 - Connecting succeeded.
2018-05-25 02:56:18 main DEBUG module8.proc.OutParamDemo:50 - ID : 2 --> Name John .
2018-05-25 02:56:18 main DEBUG module8.proc.OutParamDemo:57 - Connection closed.
```

2. Write a programme that does a wildcard search of student's name from the test.student table .
- The search criteria should be parameterized.
 - Display the matched student's id and name.
 - Handle the appropriate errors.

Database, Table

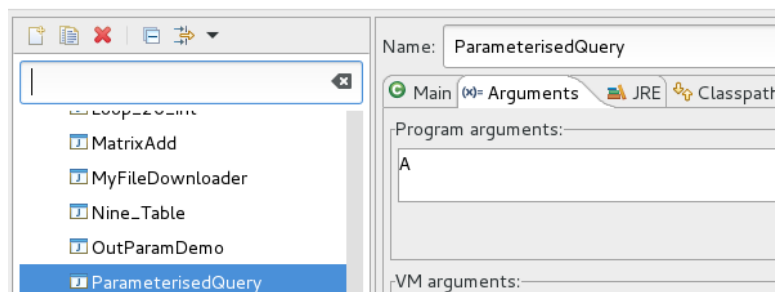
```
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0

MariaDB [test]>
MariaDB [test]>
MariaDB [test]> select * from STUDENT;
+----+-----+-----+
| ID | NAME | MARKS |
+----+-----+-----+
| 1  | Tom  | 95     |
| 2  | John | 75     |
| 3  | Alice| 90     |
| 4  | Bob  | 98     |
| 5  | Anna | 96     |
| 6  | Alex | 77     |
| 7  | Aaron| 82     |
| 8  | Amy  | 93     |
+----+-----+-----+
8 rows in set (0.00 sec)
```

Input Arguments used for the search

Create, manage, and run configurations

Run a Java application



Console Log4j Output

```
Problems Console Javadoc Declaration Terminal Debug
<terminated> ParameterisedQuery [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:38 - Connecting to database.
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:41 - Connecting succeeded.
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:50 - 3 Alice 90
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:50 - 5 Anna 96
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:50 - 6 Alex 77
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:50 - 7 Aaron 82
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:50 - 8 Amy 93
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:52 - Fetched 5 rows.
2018-05-25 03:25:05 main DEBUG module8.query.ReadingTableDemo:58 - Connection closed.
```

Java Code

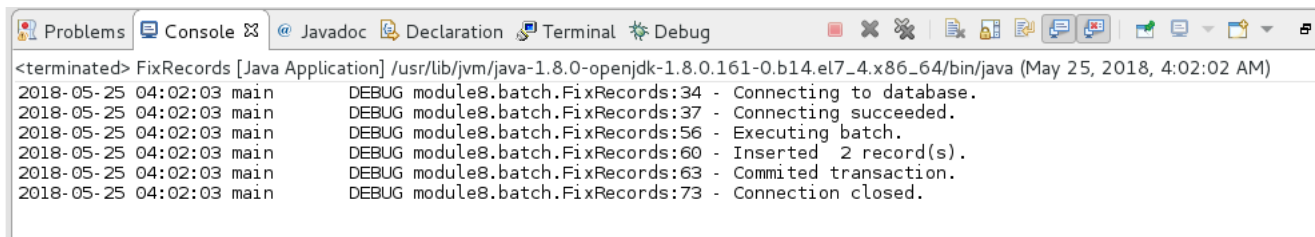
```
ParameterisedQuery.java OutParamDemo.java TestConnectionMySQL.java
5 import java.io.IOException;
6 import java.sql.Connection;
7 import java.sql.DriverManager;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import java.sql.Statement;
11 import java.util.Properties;
12
13 import org.apache.log4j.Logger;
14
15 public class ParameterisedQuery {
16
17     private static final Logger log4j = Logger.getLogger(ReadingTableDemo.class.getName());
18     private static final String DB_CONN_FILE = "resources/dbconn.properties";
19
20     public static void main(String[] args) {
21
22         if (args.length != 1) {
23             System.out.println("Usage: java ParameterisedQuery One_Character");
24             System.exit(-1);
25         }
26         String paramSearch = args[0];
27
28         Properties p = new Properties();
29
30         try {
31             p.load(new FileInputStream(new File(DB_CONN_FILE)));
32         } catch (IOException e) {
33             log4j.error("Error reading " + DB_CONN_FILE, e);
34             System.exit(-1);
35         }
36
37         log4j.debug("Connecting to database.");
38
39         try (Connection conn = DriverManager.getConnection(p.getProperty("url"), p);) {
40             log4j.debug("Connecting succeeded.");
41
42             String sqlStat = "select * from STUDENT WHERE NAME like '" + paramSearch + "%'";
43             Statement stmt = conn.createStatement();
44
45             try (ResultSet set = stmt.executeQuery(sqlStat);) {
46                 int counter = 0;
47                 while (set.next()) {
48                     counter++;
49                     log4j.debug(set.getString("ID") + " " + set.getString("NAME") + " " + set.getString("MARKS"));
50                 }
51                 log4j.debug("Fetched " + counter + " rows.");
52             }
53
54         } catch (SQLException e) {
55             log4j.error("Query failed.", e);
56         } finally {
57             log4j.debug("Connection closed.");
58         }
59     }
60 }
61
62
63 }
```

3. Write a programme to batch process insert and update statements to the test.student table.
- Handle the appropriate errors.
 - Use save-points to define boundaries of transactions.

Batch Insert

```
MariaDB [test]> select * from STUDENT;
+-----+-----+-----+
| ID | NAME | MARKS |
+-----+-----+-----+
| 1 | Tom | 95 |
| 2 | John | 75 |
| 3 | Alice | 90 |
| 4 | Bob | 98 |
| 5 | Anna | 96 |
| 6 | Alex | 77 |
| 7 | Aaron | 82 |
| 8 | Amy | 93 |
+-----+-----+-----+
8 rows in set (0.00 sec)

MariaDB [test]> select * from STUDENT;
+-----+-----+-----+
| ID | NAME | MARKS |
+-----+-----+-----+
| 1 | Tom | 95 |
| 2 | John | 75 |
| 3 | Alice | 90 |
| 4 | Bob | 98 |
| 5 | Anna | 96 |
| 6 | Alex | 77 |
| 7 | Aaron | 82 |
| 8 | Amy | 93 |
| 9 | Bella | 65 |
| 10 | Bailly | 75 |
+-----+-----+-----+
10 rows in set (0.00 sec)
```



The screenshot shows an IDE console window with the following tabs: Problems, Console, Javadoc, Declaration, Terminal, and Debug. The console output displays the execution of a Java application named 'FixRecords'.

```
<terminated> FixRecords [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 25, 2018, 4:02:02 AM)
2018-05-25 04:02:03 main          DEBUG module8.batch.FixRecords:34 - Connecting to database.
2018-05-25 04:02:03 main          DEBUG module8.batch.FixRecords:56 - Executing batch.
2018-05-25 04:02:03 main          DEBUG module8.batch.FixRecords:60 - Inserted 2 record(s).
2018-05-25 04:02:03 main          DEBUG module8.batch.FixRecords:63 - Committed transaction.
2018-05-25 04:02:03 main          DEBUG module8.batch.FixRecords:73 - Connection closed.
```

Java Code

```
10 import java.sql.Savepoint;
11 import java.util.Properties;
12
13 import org.apache.log4j.Logger;
14
15 public class FixRecords {
16
17     private static final Logger log4j = Logger.getLogger(FixRecords.class.getName());
18     private static final String DB_CONN_FILE = "resources/dbconn.properties";
19
20     public static void main(String[] args) {
21         Properties p = new Properties();
22
23         try {
24             p.load(new FileInputStream(new File(DB_CONN_FILE)));
25
26         } catch (IOException e) {
27             log4j.error("Error reading " + DB_CONN_FILE, e);
28             System.exit(-1);
29         }
30
31         log4j.debug("Connecting to database.");
32
33         try (Connection conn = DriverManager.getConnection(p.getProperty("url"), p);) {
34             log4j.debug("Connecting succeeded.");
35
36             Savepoint marker = null;
37             conn.setAutoCommit(false);
38
39             String sqlStat = "INSERT INTO STUDENT (NAME, MARKS) VALUE(?, ?)"; // PK AutoIncrement used
40
41             try (PreparedStatement stmt = conn.prepareStatement(sqlStat);) {
42                 // Start transaction
43                 marker = conn.setSavepoint("start_trans");
44                 stmt.setString(1, "Bella");
45                 stmt.setInt(2, 65);
46                 stmt.addBatch();
47
48                 stmt.setString(1, "Bailly");
49                 stmt.setInt(2, 75);
50                 stmt.addBatch();
51
52                 log4j.debug("Executing batch.");
53
54                 int[] r = stmt.executeBatch();
55
56                 log4j.debug("Inserted " + r.length + " record(s).");
57
58                 conn.commit();
59                 log4j.debug("Committed transaction.");
60                 // Ending transaction
61                 conn.releaseSavepoint(marker);
62                 conn.setAutoCommit(true);
63             }
64
65         } catch (SQLException e) {
66             log4j.error("Batch processing failed.", e);
67         } finally {
68             log4j.debug("Connection closed.");
69         }
70     }
71 }
```

Update records Batch

```
File Edit View Search Terminal Help
MariaDB [test]> select * from STUDENT;
+-----+-----+-----+
| ID | NAME | MARKS |
+-----+-----+-----+
| 1 | Tom | 95 |
| 2 | John | 75 |
| 3 | Alice | 90 |
| 4 | Bob | 98 |
| 5 | Anna | 96 |
| 6 | Alex | 77 |
| 7 | Aaron | 82 |
| 8 | Amy | 93 |
| 9 | Bella | 65 |
| 10 | Bailly | 75 |
+-----+-----+-----+
10 rows in set (0.00 sec)

MariaDB [test]> select * from STUDENT;
+-----+-----+-----+
| ID | NAME | MARKS |
+-----+-----+-----+
| 1 | Tom | 95 |
| 2 | John | 75 |
| 3 | Alice | 90 |
| 4 | Bob | 98 |
| 5 | Anna | 96 |
| 6 | Alex | 77 |
| 7 | Aaron | 82 |
| 8 | Amy | 93 |
| 9 | Bella | 33 |
| 10 | Bailly | 66 |
+-----+-----+-----+
10 rows in set (0.00 sec)
```

console Log4j output

```
Problems Console Javadoc Declaration Terminal Debug
<terminated> FixRecords [Java Application] /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.161-0.b14.el7_4.x86_64/bin/java (May 25, 2018, 4:14:43 AM)
2018-05-25 04:14:44 main DEBUG module8.batch.FixRecords:31 - Connecting to database.
2018-05-25 04:14:44 main DEBUG module8.batch.FixRecords:34 - Connecting succeeded.
2018-05-25 04:14:44 main DEBUG module8.batch.FixRecords:53 - Executing batch.
2018-05-25 04:14:44 main DEBUG module8.batch.FixRecords:57 - Inserted/updated 2 record(s).
2018-05-25 04:14:44 main DEBUG module8.batch.FixRecords:60 - Committed transaction.
2018-05-25 04:14:44 main DEBUG module8.batch.FixRecords:69 - Connection closed.
```

```

5
7 private static final Logger log4j = Logger.getLogger(FixRecords.class.getName());
3 private static final String DB_CONN_FILE = "resources/dbconn.properties";
3
3- public static void main(String[] args) {
1     Properties p = new Properties();
2
3     try {
4         p.load(new FileInputStream(new File(DB_CONN_FILE)));
5
6     } catch (IOException e) {
7         log4j.error("Error reading " + DB_CONN_FILE, e);
8         System.exit(-1);
9     }
10
11     log4j.debug("Connecting to database.");
12
13     try (Connection conn = DriverManager.getConnection(p.getProperty("url"), p);) {
14         log4j.debug("Connecting succeeded.");
15
16         Savepoint marker = null;
17         conn.setAutoCommit(false);
18
19         //String sqlStat = "INSERT INTO STUDENT (NAME, MARKS) VALUE(?, ?)"; // PK AutoIncrement us
20         String sqlStat = "UPDATE STUDENT SET MARKS=? WHERE ID=?";
21
22         try (PreparedStatement stmt = conn.prepareStatement(sqlStat);) {
23             // Start transaction
24             marker = conn.setSavepoint("start_trans");
25             stmt.setInt(1, 66);
26             stmt.setInt(2, 10);
27             stmt.addBatch();
28
29             stmt.setInt(1, 33);
30             stmt.setInt(2, 9);
31             stmt.addBatch();
32
33             log4j.debug("Executing batch.");
34
35             int[] r = stmt.executeBatch();
36
37             log4j.debug("Inserted/updated " + r.length + " record(s).");
38
39             conn.commit();
40             log4j.debug("Committed transaction.");
41             // Ending transaction
42             conn.releaseSavepoint(marker);
43             conn.setAutoCommit(true);
44         }
45
46     } catch (SQLException e) {
47         log4j.error("Batch processing failed.", e);
48     } finally {
49         log4j.debug("Connection closed.");
50     }
51 }
52
53 }
54 }

```