Principles of Databases

JDBC & Stored Procedures

David Sinclair

Overview

JDBC (Java DataBase Connectivity) is an API that provides connectivity to an SQL database. It allows us to invoke SQL statements from a Java program.

The basic steps in using JDBC are:

1. Load the driver.
2. Establish a connection to a database.
3. Prepare an SQL statement.
4. Execute an SQL statement.
5. Get the **Result Set**.
6. Close the connection to the database.

Simple JDBC Program

```java
// Demonstrates how to use JDBC to query a MySQL database
// from within a Java program.

import java.io.*;
import java.util.Scanner;
import java.sql.*;

class JDBCTest
{

  public static void main (String args [])
     throws SQLException, IOException
  {
      Scanner sc = new Scanner(System.in);

    try
    {
      Class.forName("com.mysql.jdbc.Driver");
    }
    catch (ClassNotFoundException cnf)
    {
      System.out.println ("Driver could not be loaded");
      System.exit (0);
    }
```

Simple JDBC Program (2)

```java
    String dbacct, passwrd, min_score, max_score, sname,
           uname, degree;
    Integer sid, score;

    System.out.print ("Enter database account: ");
    dbacct = sc.next();

    Connection conn = DriverManager.getConnection
                        ("jdbc:mysql://localhost:3306/" + dbacct,
                         "root","");

    String stmt1 = "select sID, sName, score, degree, uName " +
                   "from Student join Apply using (sID) " +
                   "where score >= ?" +
                   " and score <= ? and degree = ?";
    PreparedStatement p = conn.prepareStatement(stmt1);

    System.out.print("Enter minimum score: ");
    min_score = sc.next();
    System.out.print("Enter maximum score: ");
    max_score = sc.next();
    System.out.print("Enter degree: ");
    degree = sc.next();
```

Simple JDBC Program (3)

```
48          p.clearParameters();
49          p.setString(1, min_score);
50          p.setString(2, max_score);
51          p.setString(3, degree);
52
53          ResultSet r = p.executeQuery();
54
55          System.out.println();
56          while (r.next())
57          {
58            sid = r.getInt(1);
59            sname = r.getString(2);
60            score = r.getInt(3);
61            degree = r.getString(4);
62            uname = r.getString(5);
63            System.out.println(sid + "␣" + sname + "␣" + score +
64                               "␣" + degree + "␣" + uname);
65          }
66          System.out.println();
67        }
68
69        conn.close ();
70      }
```

Notes on the Program

- Line 18 loads the JDBC driver.
- Lines 36-40 prepare an SQL statement with placeholders for variables represented by the '?' characters.
- Lines 49-51 uses `setString` set the first, second and third placeholders with the values stored in min_score, max_score and degree respectively.
- Line 53 executes the the final query and returns a **ResultSet**.
- In this example, each element of the **ResultSet** has 5 values. Lines 59-62 use `getInt` and `getString` to extract the values from each element of the **ResultSet**.
- Line 69 closes the connection to the mySQL database.

Stored Procedures

mySQL, as well as other SQL systems, allow the user to store SQL code in the database server. It can be invoked by a program, a trigger or another stored procedure. It can perform some of generic task, usually business logic, which is dependent to database table data.

They have several advantages.

- If multiple applications are running in multiple environments and need to use the same database, then by using a stored procedure you can make your business logic independent of the programming language.
- A stored procedure may increases the performance of the application. When stored procedure is created, it is compiled. When it is called, it does not go through the parser and is directly executed.
- It is reusable and the user does not need to know the database structure.

Stored Procedures (2)

They also have several disadvantages.

- Stored procedures very hard to debug. This is mainly because they are declarative, through there are some selection and looping constructs.
- Managing stored procedure is difficult because they do not have any structure like object oriented programs.
- Mysql stored procedures never return a value directly. Either you need to specify output parameter in your stored procedure or put select statement inside the procedure.

Creating a Stored Procedure

```
1 DELIMITER //
2 CREATE PROCEDURE GetAllStudents()
3     BEGIN
4     SELECT * FROM Student;
5     END //
6 DELIMITER ;
```

- Since a stored procedure may have several SQL commands, each terminated with a semicolon, we need to define a new delimiter (line 1). Line 6 restores the default delimiter.
- The section between BEGIN and END is called the body of the stored procedure. You put the declarative SQL statements in the body to handle the business logic.
- Stored procedures are not case sensitive.
- To invoke the above stored procedure, you type `CALL GetAllStudents;`

Stored Procedure Variables

You can declare a variable inside a stored procedure by:

```
DECLARE variable_name datatype(size) DEFAULT default_value;
```

- You need specify the data type of the variable and its size. A variable can have any MySQL data types such as INT, VARCHAR, etc.
- When you declare a variable, its initial value is NULL. You can assign the variable a default value by using DEFAULT keyword.

To assign a value to a stored procedure variable, `TotalStudents` use either

```
SET TotalStudents = 0;
```

or

```
SELECT COUNT(*) INTO TotalStudents FROM Student;
```

Scope of Stored Procedure Variables

A variable declared inside a BEGIN END block, it will be out of scope when the END is reached. You can declare two or more variables with the same name in different scopes because a variable is only effective in its own scope.

A variable that begins with the @ sign at the beginning is a session variable. It is available and accessible until the session ends.

MySQL Stored Procedure Parameters

A stored proedure parameter has one of three modes IN, OUT or INOUT.

IN
: This is the default mode. When you define an IN parameter in a stored procedure, the calling program has to pass an argument to the stored procedure. The stored procedure only works on the copy of the IN parameter, so its original value is retained after the stored procedure ends.

OUT
: The value of an OUT parameter can be changed inside the stored procedure and its new value is passed back to the calling program.

INOUT
: An INOUT parameter is the combination of IN parameter and OUT parameter.

MySQL Stored Procedure Parameters

```
DELIMITER //
CREATE PROCEDURE CountByCountry(IN CountryName VARCHAR(25),
                                OUT total INT)
BEGIN
    SELECT count(*)
    INTO total
    FROM Person, County
    WHERE Person.countryID = Country.countryID AND
          Country.Name = CountryName;
END//
DELIMITER ;
```

and this could be invoked by

```
CALL CountByCountry ("China", @totalNumber);
SELECT @totalNumber;
```

Selection and Loops

The **WHILE** statement

```
DELIMITER //
DROP PROCEDURE IF EXISTS WhileLoopProc//
CREATE PROCEDURE WhileLoopProc()
       BEGIN
               DECLARE x  INT;
               DECLARE str  VARCHAR(255);
               SET x = 1;
               SET str =  '';
               WHILE x  <= 5 DO
                           SET  str = CONCAT(str,x,',');
                           SET  x = x + 1;
               END WHILE;
               SELECT str;
       END//
DELIMITER ;
```

Selection and Loops (2)

The **REPEAT** statement

```
DELIMITER //
DROP PROCEDURE IF EXISTS RepeatLoopProc//
CREATE PROCEDURE RepeatLoopProc()
       BEGIN
               DECLARE x  INT;
               DECLARE str  VARCHAR(255);
               SET x = 1;
               SET str =  '';
               REPEAT
                           SET  str = CONCAT(str,x,',');
                           SET  x = x + 1;
               UNTIL x  > 5
               END REPEAT;
               SELECT str;
       END//
DELIMITER ;
```

- Notice there is no semicolon after the `UNTIL` statement.

Selection and Loops (3)

The **LOOP, LEAVE, ITERATE** and **IF** statements

```
DELIMITER //
DROP PROCEDURE IF EXISTS LOOPLoopProc//
CREATE PROCEDURE LOOPLoopProc()
     BEGIN
           DECLARE x  INT;
           DECLARE str  VARCHAR(255);
           SET x = 1;
           SET str =  '';
           loop_label:  LOOP
                            IF  x > 10 THEN
                               LEAVE  loop_label;
                            END  IF;
                            SET  x = x + 1;
                            IF  (x mod 2) THEN
                               ITERATE  loop_label;
                            ELSE
                               SET  str = CONCAT(str,x,',');
                            END  IF;
                       END LOOP;
             SELECT str;
     END//
DELIMITER ;
```

mySQL Cursor

In order to process a result set inside a store procedure you need to use a mySQL cursor.

```
DELIMITER //
CREATE PROCEDURE build_list (INOUT capital_list varchar(400))
BEGIN
    DECLARE v_finished INTEGER DEFAULT 0;
    DECLARE v_capital varchar(20) DEFAULT "";
    DECLARE capital_cursor CURSOR FOR
        SELECT Capital FROM Country;
     DECLARE CONTINUE HANDLER
        FOR NOT FOUND SET v_finished = 1;
    OPEN capital_cursor;
    get_capital: LOOP
        FETCH capital_cursor INTO v_capital;
        IF v_finished = 1 THEN
            LEAVE get_capital;
        END IF;
        SET capital_list = CONCAT(v_capital,";",capital_list);
    END LOOP get_capital;
    CLOSE capital_cursor;
END//
DELIMITER ;
```