Practices for Lesson 18: Building Database Applications with JDBC

Chapter 18

Practices for Lesson 18: Overview

Practices Overview

In these practices, you will work with the JavaDB (Derby) database, creating, reading, updating, and deleting data from a SQL database by using the Java JDBC API.

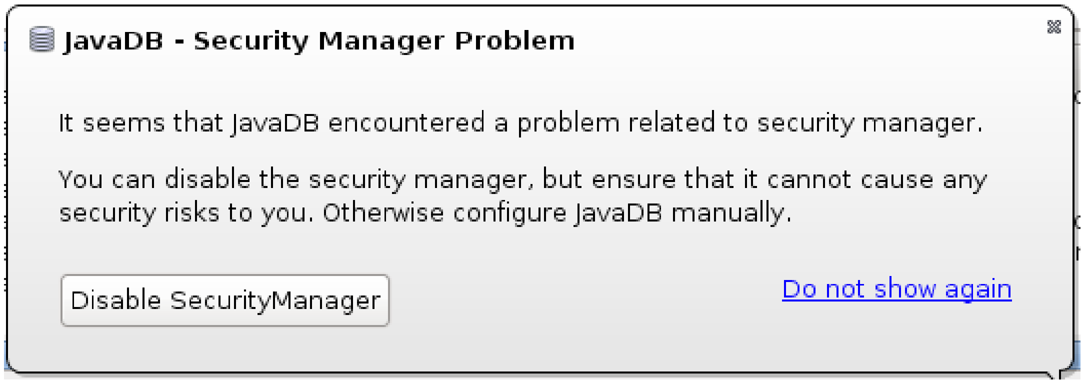
Practice 18-1: Summary Level: Working with the Derby Database and JDBC

Overview

In this practice, you will start the JavaDB (Derby) database, load some sample data using a script, and write an application to read the contents of an employee database table and print the results to the console.

Note

The first time you run the JavaDB server you will get the following error message:



JavaDB - Security Manager Problem

It seems that JavaDB encountered a problem related to security manager.

You can disable the security manager, but ensure that it cannot cause any security risks to you. Otherwise configure JavaDB manually.

Go ahead and click the **Disable SecurityManager** button. This will enable you to complete the practices.

Tasks

1. Create the Employee Database by using the SQL script provided in the resource directory. Perform the following steps in NetBeans:
   a. Click the Services tab.
   b. Expand the Databases folder.
   c. Right-click JavaDB and select Start Server.
   d. Right-click JavaDB again and select Create Database.
   e. Enter the following information:

Window/Page Description	Choices or Values
Database Name	EmployeeDB
User Name	tiger
Password	scott
Confirm Password	scott

   f. Click OK
   g. Right-click the connection that you created: `jdbc:derby://localhost:1527/EmployeeDB[tiger on TIGER]` and select Connect.
   h. Select File > Open File.
   i. Browse to `/home/oracle/labs/resources/EmployeeTable.sql` script. The file will open in a SQL Execute window.

j. Select the connection that you created from the drop-down list, and click the Run-SQL icon or press Ctrl-Shift-E to run the script.

k. Expand the `EmployeeDB` connection. You will see that the `TIGER` schema is now created. Expand the `TIGER` Schema and look at the table `Employee`.

l. Right-click the connection again and select Execute Command to open another SQL window. Enter the command:
```
select * from Employee
```
and click the Run-SQL icon to see the contents of the `Employee` table.

2. Open the `SimpleJDBC18-01Prac` project and run it.

   a. You should see all the records from the Employee table displayed.

   Note: In case you get a broken reference link to Java DB driver error, perform the following steps:

      I. Right-click on the project and select properties.
      II. In the categories column select Libraries.
      III. Click Add Library and select Java DB Driver from the Available libraries
      IV. Click Add Library.
      V. Click OK.

3. Add a SQL command to add a new Employee record.

   a. Modify the `SimpleJDBCExample` class to add a new Employee record to the database.

   Note: If you run the application again, it will throw an exception, because this key already exists in the database.

   b. The syntax for adding a row in a SQL database is:
```
INSERT INTO <table name> VALUES (<column 1 value>, <column 2
value>, ...)
```

   c. Use the Statement `executeUpdate` method to execute the query. What is the return type for this method? What value should the return type be? Test to make sure that the value of the return is correct.

Practice 18-1: Detailed Level: Working with the Derby Database and JDBC

Overview

In this practice, you will start the JavaDB (Derby) database, load some sample data using a script, and write an application to read the contents of an employee database table and print the results to the console.

Tasks

1. Create the Employee Database by using the SQL script provided in the resource directory. Perform the following steps in NetBeans:

   a. Click Services tab.
   b. Expand the Databases folder.
   c. Right-click JavaDB and select Start Server.
   d. Right-click JavaDB again and select Create Database.
   e. Enter the following information:

Window/Page Description	Choices or Values
Database Name	EmployeeDB
User Name	tiger
Password	scott
Confirm Password	scott

   f. Click OK.
   g. Right-click the connection that you created: `jdbc:derby://localhost:1527/EmployeeDB[tiger on TIGER]` and select Connect.
   h. Select File > Open File.
   i. Browse to `/home/oracle/labs/resources` and open the `EmployeeTable.sql` script. The file will open in a SQL Execute window.
   j. Select the connection that you created from the drop-down list and click the Run-SQL icon or press Ctrl-Shift-E to run the script.
   k. Expand the `EmployeeDB` connection. You will see that the `TIGER` schema is now created. Expand the `TIGER` Schema, expand Tables, and then expand the table `Employee`.
   l. Right-click the connection again and select Execute Command to open another SQL window. Enter the command:
   `select * from Employee`
   and click the Run-SQL icon to see the contents of the `Employee` table.

2. Open the `SimpleJDBC18-01Prac` Project and run it.
   a. Select File > Open Project.
   b. Select `/home/oracle/labs/18-JDBC/practices/practice1/SimpleJDBC18-01Prac`.
   c. Click Open Project.
   d. Expand the Source Packages and look at the `SimpleJDBCExample.java`
   e. Run the project: Right-click the project and select Run, or click the Run icon, or press F6.
   f. You should see all the records from the Employee table displayed.

Note: In case you get a broken reference link to Java DB driver error, perform the following steps:

VI. Right-click on the project and select properties.
VII. In the categories column select Libraries.
VIII. Click Add Library and select Java DB Driver from the Available libraries.
IX. Click Add Library.
X. Click OK.

3. Add a SQL command to add a new Employee record.
   a. Modify the `SimpleJDBCExample` class to add a new Employee record to the database.
   b. The syntax for adding a row in a SQL database is:
      `INSERT INTO <table name> VALUES (<column 1 value>, <column 2 value>, ...)`
   c. Use the Statement `executeUpdate` method to execute the query. What is the return type for this method? What value should the return type be? Test to make sure that the value of the return is correct.
   d. Your code may look like this:

```
query = "INSERT INTO Employee VALUES (400, 'Bill',
'Murray','1950-09-21', 150000)";
if (stmt.executeUpdate(query) != 1) {
    System.out.println ("Failed to add a new employee record");
}
```

Note: If you run the application again, it will throw an exception, because this key already exists in the database.