UNIT 15: DATABASE PROGRAMMING USING JAVA

UNIT STRUCTURE

15.1 Learning Objectives

15.2 Introduction

15.3 MySQL as Backend and Java As Frontend

- 15.3.1 JDBC Steps to Connect MySQLDatabase
- 15.3.2 Extract Data from MySQL Database
- 15.3.3 Create MySQLTable Using JDBC
- 15.3.4 Insert Data into Table
- 15.3.5 Update Data into Table
- 15.3.6 Delete Data from Table
- 15.3.7 Application of Java Swing and MySQL

15.4 Let Us Sum Up

15.5 Further Reading

15.6 Answer to Check Your Progress

15.7 Model Questions

15.1 LEARNING OBJECTIVES

After going through this unit, you will be able to:

- connect JDBC with MySQL
- explain how to create database in MySQL
- explain how to extract data from MySQL table
- describe how to create table in MySQL through java programming
- describe how to insert and update data to MySQL table
- explain how to delete data from MySQL table
- describe application of user input though swing component and store into MySQL table.

15.2 INTRODUCTION

In the earlier unit, you have been introduced to JDBC. Topics like JDBC API and JDBC drivers along with the design and structure of JDBC have been described in the previous unit. In addition to these, concepts related to two-tier and three-tier architecture and ResultSet were also described in the previous unit.

This is the last unit of the course and in this unit, you will be introduced to database programming using JAVA. Topics like connecting to MySQL database and extracting data from MySQL databases are described in this unit. In addition to this, concepts related to creating a table, inserting, deleting and updating data from a table are also described in this unit.

15.3 MYSQL AS BACKEND AND JAVA AS FRONTEND

Practically MySQL is one of the RDBMS.DriverManager have JDBC driver which try to make Connections with MySQL database. The easiest way to connect is to use Class.forName() on the class that implements the java.sql.Driver interface. With MySQL Connector, the name of this class is com.mysql.jdbc.Driver. With this method, you could use an external configuration file to supply the driver class name and driver parameters to use when connecting to a database.

15.3.1 JDBC Steps to Connect MySQL Database

To connect Java application with the MySQL database, we need to go through the following steps.

- **Driver class:** The driver class com.mysql.jdbc.Driver use for the mysql database.
- **Connection URL:** The connection URL for the mysql database is jdbc:mysql://localhost:3306/DB where jdbc is the API, mysql is the database, localhost is the server name where mysql is running, 3306 is the port number and DB is the database name.

- **Username:** The default username for the mysql database is root.
- **Password:** The password choosen by the user at the time of installing the mysql database.

In MySQL, first create the database before creating table, to create database and table steps are as follows:

How to create database and table in MySQL

```
1. create database DB;
2. use DB;
3. create table emp(id int(10),name varchar(40),age int(3));
```

Example 15.1: Example to open and close connected database

```
import java.sql.Connection;
import java.sql.DriverManager;
public class Connectiondata{
public static void main(String[] argv) throws Exception{
String driver = "com.mysql.jdbc.Driver";
String connection = "jdbc:mysql://localhost:3306/DBName";
String user = "root";
String password = "root";
Class.forName(driver);
Connection con = DriverManager.getConnection (connection, user, password);
if(!con.isClosed()) {
con.close();
}
}
}
```

To connect java application with the mysql database, mysqlconnector.jar file is required to be loaded.

Two ways to load the jar file:

1. Paste the mysqlconnector.jar file in jre/lib/ext folder
2. Set classpath

1. Paste the mysqlconnector.jar file in JRE/lib/ext folder:

Download the mysqlconnector .jar file and paste the jar file in jre/lib/ext folder

2. Set classpath:

Set the classpath in a temporary or permanent way.

15.3.2 Extract Data FromMySQL Database

Example 15.2: Create a Connection of Java Application to MySQL database and read data from MySQL table data and display on the output screen.

```
import java.sql.*;
class MysqlConnect{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/DB","root","root");
//here root is username and password
Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+"
"+rs.getString(2)+"  "+rs.getString(3));
con.close();
}catch(Exception e)
{ System.out.println(e);}
}
}
```

15.3.3 Create MySQL Table Using JDBC

Example 15.3: Example to create MySQL table with java application

```
package Edu;
import java.sql.*;
import java.sql.DriverManager;
public class Example {
static final String JDBC_DRIVER =
"com.mysql.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://
localhost/emp";
static final String USER = "root";
static final String PASS = "";
public static void main(String[] args) {
Connection conn = null;
Statement stmt = null;
try{
//STEP 2: Register JDBC driver
Class.forName("com.mysql.cj.jdbc.Driver");
System.out.println("Connecting to database...");
conn =
DriverManager.getConnection(DB_URL,"root","");
System.out.println("Creating table in given
database...");
stmt = conn.createStatement();
String sql = "CREATE TABLE EMPLOYEES " +
"(id INTEGER not NULL, " +
" first VARCHAR(255), " +
" last VARCHAR(255), " +
" age INTEGER, " +
" PRIMARY KEY ( id ))";
stmt.executeUpdate(sql);
System.out.println("Created table in given
```

```
database...");
System.out.println("Inserting records into the
table...");
stmt =conn.createStatement();
String sql ="INSERT INTO Employees VALUES (100,
'Kriss', 'Kurian', 18)";
stmt.executeUpdate(sql);
sql = "INSERT INTO Employees VALUES (101, 'Enrique',
'John', 25)";
stmt.executeUpdate(sql);
sql= "INSERT INTO Employees (102, 'Taylor', 'Swift',
30)";
stmt.executeUpdate(sql);
sql= "INSERT INTO Employees VALUES(103, 'Linkin',
'Park', 28)";
stmt.executeUpdate(sql);
System.out.println("Inserted records into the
table...");
rs.close();
stmt.close();
conn.close();
}catch(SQLException se){
se.printStackTrace();
}catch(Exception e){
e.printStackTrace();
}finally{
try{
if(stmt!=null)
stmt.close()
}catch(SQLException se2){
}               // nothing can be done
try{
if(conn!=null)
```

```
conn.close();
}catch(SQLException se){
se.printStackTrace();
}
}
System.out.println("Goodbye!");
}
}
}
```



CHECK YOUR PROGRESS

Q 1. Describe the steps to establish JDBC connection.

Q 2. How can you create a table in MYSQL?

Q 3. How can you create an object for connection?

Q 4. Write SQL statement for insert data into table.

15.3.4 Insert Data Into Table

Example 15.4: Example of inserting Data into a MySQL table using PreparedStatement

```
import java.sql.*;
import java.util.Calender;
public class JavaMysqlPrStatementInsert{
public static void main(String[] args)
{ try{
String myDriver = "org.gjt.mm.mysql.Driver";
String myUrl = "jdbc:mysql://localhost/test";
Class.forName(myDriver);
Connection conn =
DriverManager.getConnection(myUrl, "root", "");
Calendar calendar = Calendar.getInstance();
```

```
java.sql.DatestartDate = new
java.sql.Date(calendar.getTime().getTime());
String query = " insert into users (first_name,
last_name, date_created, is_admin, num_points)"+"
values(?, ?, ?, ?, ?)";
PreparedStatementpreparedStmt =
conn.prepareStatement(query);
preparedStmt.setString (1, "Barney");
preparedStmt.setString (2, "Rubble");
preparedStmt.setDate (3, startdate);
preparedStmt.setBoolean(4, false);
preparedStmt.setInt (5, 5000);
preparedStmt.execute();
conn.close();
}
Catch( Exception e)
{
System.err.println("Got an exception!");
System.err.println(e.getMessage());
}
}
}
```

15.3.5 Update Data Into Table

Example 15.5: Example of Update Data into a MySQL table using PreparedStatement

```
Import java.sql.*;
Import java.util.Calender;
Public class JavaMysqlPrStatementUpdate{
public static void main(String[] args)
{ try{
String myDriver = "org.gjt.mm.mysql.Driver";
```

```
String myUrl = "jdbc:mysql://localhost/test";
Class.forName(myDriver);
Connection conn =
DriverManager.getConnection(myUrl, "root", "");
String query = "update users set num_points= ?
where first_name = ?";
PreparedStatementpreparedStmt =
conn.prepareStatement(query);
preparedStmt.setInt (1, 6000);
preparedStmt.setString (2, "XYZ");
preparedStmt.executeUpdate();
conn.close();
}
Catch( Exception e)
{
System.err.println("Got an exception!");
System.err.println(e.getMessage());
}
}
}
```

15.3.6 Delete Data From Table

Example 15.6: Example to delete record from table in MySQL Database using PreparedStatement

```
package com.boraji.tutorial.jdbc;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.PreparedStatement;
importjava.sql.SQLException;
publicclassDeleteRecordExample{
publicstaticvoidmain(String[] args) {
String jdbcUrl = "jdbc:mysql://localhost:3306/DB";
```

```
String username = "root";
String password = "root";
String sql = "delete from employee where emp_id=?";
try
(Connection conn =
DriverManager.getConnection(jdbcUrl,username,
password);
PreparedStatementstmt = conn.prepareStatement(sql))
{ stmt.setInt(1, 2);
stmt.executeUpdate();
System.out.println("Record deleted successfully");
}
catch (SQLException e)
{ e.printStackTrace();
 }
}
}
```

15.3.7 Application Of Java Swing And MySQL

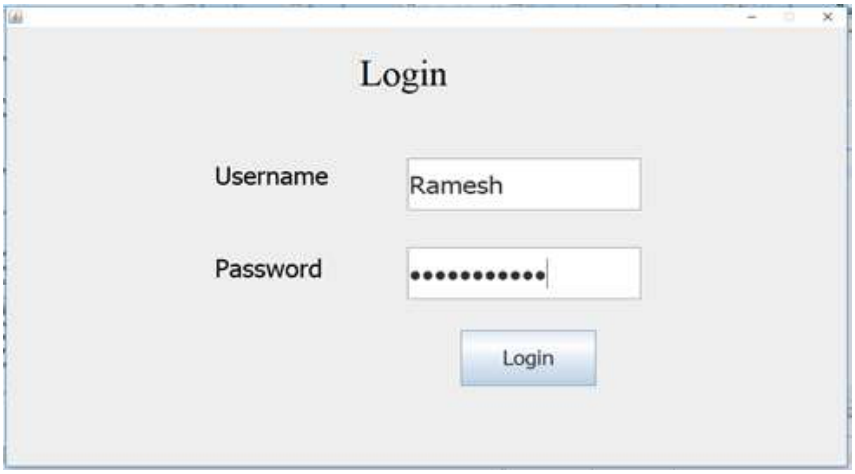


Figure 15.1: Example of user login page

Program 15.1: Create a simple java program to user login form using Swing APIs.

```
packagecom.javaguides.javaswing.login;
importjava.awt.Color;
importjava.awt.EventQueue;
importjava.awt.Font;
importjava.awt.event.ActionEvent;
importjava.awt.event.ActionListener;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.PreparedStatement;
importjava.sql.ResultSet;
importjava.sql.SQLException;
importjavax.swing.JButton;
importjavax.swing.JFrame;
importjavax.swing.JLabel;
importjavax.swing.JOptionPane;
importjavax.swing.JPanel;
importjavax.swing.JPasswordField;
importjavax.swing.JTextField;
importjavax.swing.border.EmptyBorder;
publicclassUserLoginextendsJFrame {
privatestaticfinallongserialVersionUID=1L;
privateJTextFieldtextField;
privateJPasswordFieldpasswordField;
privateJButtonbtnNewButton;
privateJLabel label;
privateJPanelcontentPane;
publicstaticvoidmain(String[] args) {
EventQueue.invokeLater(newRunnable()  {
publicvoidrun() {
try  {
```

```
        UserLogin frame =newUserLogin();
        frame.setVisible(true);
        } catch (Exception e) {
        e.printStackTrace();
        }
        }
        });
        }
        publicUserLogin()     {

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setBounds(450, 190, 1014, 597);
        setResizable(false);
        contentPane=newJPanel();
        contentPane.setBorder(newEmptyBorder(5, 5, 5,
5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        JLabellblNewLabel=newJLabel("Login");
        lblNewLabel.setForeground(Color.BLACK);
        lblNewLabel.setFont(newFont("Times New
Roman", Font.PLAIN, 46));
        lblNewLabel.setBounds(423, 13, 273, 93);
        contentPane.add(lblNewLabel);
        textField=newJTextField();
        textField.setFont(newFont("Tahoma",
Font.PLAIN, 32));
        textField.setBounds(481, 170, 281, 68);
        contentPane.add(textField);
        textField.setColumns(10);
        passwordField=newJPasswordField();
```

```
textField.setFont(newFont("Tahoma",
Font.PLAIN, 32));
textField.setBounds(481, 170, 281, 68);
contentPane.add(passwordField);
JLabellblUsername=newJLabel("Username");
lblUsername.setBackground(Color.BLACK);
lblUsername.setForeground(Color.BLACK);
lblUsername.setFont(newFont("Tahoma",
Font.PLAIN, 31));
lblUsername.setBounds(250, 166, 193, 52);
contentPane.add(lblUsername);
JLabellblUsername=newJLabel("Password");
lblUsername.setBackground(Color.BLACK);
lblUsername.setForeground(Color.CYAN);
lblUsername.setFont(newFont("Tahoma",
Font.PLAIN, 31));
lblUsername.setBounds(250, 166, 193, 52);
contentPane.add(lblPassword);
btnNewButton=newJButton("Login");
btnNewButton.setFont(newFont("Tahoma",
Font.PLAIN, 26));
btnNewButton.setBounds(545, 392, 162, 73);
btnNewButton.addActionListener(newActionListener()
{
publicvoidactionPerformed(ActionEvente)
{
StringuserName=textField.getText();
String password =passwordField.getText();
try  {
Connectionconnection= (Connection)
DriverManager.getConnection("jdbc:mysql://
localhost:3306/swing_demo" "root", "root");
```

```
      PreparedStatementst=  (PreparedStatement)
connection
      .prepareStatement("Select name, password from
student where name=? and password=?");
      st.setString(1, userName);
      st.setString(2, password);
      ResultSetrs=st.executeQuery();
      if (rs.next())  {
      dispose();
      UserHome ah =newUserHome(userName);
      ah.setTitle("Welcome");
      ah.setVisible(true);
      JOptionPane.showMessageDialog(btnNewButton,
"You have successfully logged in");
      }    else {
      JOptionPane.showMessageDialog(btnNewButton,
"Wrong Username & Password");
      }
      }
      catch (SQLExceptionsqlException) {
      sqlException.printStackTrace();
      }
      }
      });
      contentPane.add(btnNewButton);
      label =newJLabel("");
      label.setBounds(0, 0, 1008, 562);
      contentPane.add(label);
      }
      }
```



CHECK YOUR PROGRESS

Q 5. Write SQL query toupdate data into MySQL table?

Q 6. Write SQL query to delete data from MySQL table?

Q 7. What is the use of the following code?

```
btnNewButton.setFont(newFont("Tahoma", Font.PLAIN,
26))
```



15.4 LET US SUM UP

- To connect Java application with the MySQL database, we need to go through the following parts: driver, connection URL, user and password.
- We can create table in MySQL using Java programming.
- We can insert, update data into MySQL table using java coding.
- Deletion of record is possible through JBDC connectivity.
- Different component like java swing also applicable to taking the user input and store data to database table.



15.5 FURTHER READING

1) Kumar, S. K.et al (2014), JDBC, Servlet, and JSP includes JSF and Design Patterns Black Book, DreamTech Press
2) Sharan, K.et al.(2022), Beginning Java 8 APIs, Extensions and Libraries, 1st ed., Apress Berkeley, CA



15.6 ANSWERS TO CHECK YOUR PROGRESS

Ans to Q No 1. The steps to establish JDBC connection are:

- define driver
- define connection URL
- username
- password

Ans to Q No 2. The steps to create MySQL table are:

```
create database DB;
use DB;
create table emp(id int(10),name varchar(40),age int(3));
```

Ans to Q No 3. The query for creating an object for connection is:

```
String sql ="INSERT INTO Employees VALUES (100,
'Kriss', 'Kurian', 18)";
```

Ans to Q No 4. The SQL statements for inserting data into table are:

```
String myDriver = "org.gjt.mm.mysql.Driver";
String myUrl = "jdbc:mysql://localhost/test";
Class.forName(myDriver);
Connection conn = DriverManager.getConnection(myUrl,
"root", "");
```

Ans to Q No 5. The string query to update data into MySQL table is:

```
"update users set num_points= ? where first_name =
?";
```

Ans to Q No 6. The query to delete data from MySQL table is:

```
String sql = "delete from employee where emp_id=?";
```

Ans to Q No 7. The use of the code is to set font properties for a button.



15.7 MODEL QUESTIONS

Q 1. Write how to create JDBC connectivity.

Q 2. Write about the important points related to the JDBC connection.

Q 3. Write MySQL coding to create a table and display a record from the table.

Q 4. Write a Java program to create tables related to your student information at your university.

Q 5. Write a Java program to extract data from the student table.

Q 6. Write a Java program to enter new data into your student table using java swing.

Q 7. Write a Java program to enter updated data into your student table from user input.

Q 8. Write a Java program to create tables to store exam-related information for students.

Q 9. Write a Java program to enter delete records from your student after passing out of the particular student.

*** ***** ***