Basic database operations using JDBC

Java Database Connectivity

JDBC Concepts and Terminology

- Java Database Connectivity (JDBC)
- Understanding the core concepts and terminology
- Role of SQL in JDBC
- JDBC Architecture
- Example JDBC PROGRAM
- Summary

What is JDBC?

- JDBC is an API in Java for connecting and executing queries on a database.
- Provides a standard interface for database-independent applications.
- Works as a bridge between Java applications and databases.

JDBC (Java Database Connectivity) is a Java API for connecting to relational databases.

Allows Java applications to execute SQL statements.

Provides a bridge between Java code and database systems.

Role of SQL in JDBC

- JDBC uses SQL statements to interact with databases.
- SQL commands are executed through JDBC objects such as Statement and PreparedStatement.
- JDBC sends SQL queries to the database and retrieves results.

Introducing SQL (Structured Query Language)

SQL is used to manage and manipulate relational databases.

Common SQL operations:

- SELECT: Retrieve data
- INSERT: Add new records
- UPDATE: Modify data
- DELETE: Remove data

Database Tables

- Tables store data in rows and columns.
- Each column has a data type and name.
- Example:

```
CREATE TABLE students (
id INT PRIMARY KEY,
name VARCHAR(50),
age INT);
```

Creating Tables using JDBC

Example:

```
Statement stmt = con.createStatement();

stmt.executeUpdate("CREATE TABLE students (id
INT, name VARCHAR(50))");
```

Inserting Data using JDBC

- Example:

```
String sql = "INSERT INTO students VALUES (1,
'Ali', 22)";
stmt.executeUpdate(sql);
```

- Adds a new record into the database table

Retrieving Data (SELECT)

- Example:

```
ResultSet rs = stmt.executeQuery("SELECT *
FROM students");
while(rs.next()) {
System.out.println(rs.getInt("id") + ", " +
rs.getString("name"));}
```

Updating and Deleting Data

- Update Example:

```
stmt.executeUpdate("UPDATE students SET age
= 23 WHERE id = 1");
```

- Delete Example:

```
stmt.executeUpdate("DELETE FROM students
WHERE id = 1");
```

JDBC Architecture

- ★ Import JDBC packages
- ★ Register the JDBC driver
- ★ Establish a database connection
- ★ Create a statement object
- ★ Execute SQL queries
- ★ Process results
- ★ Close the connection

Import JDBC Packages

```
import java.sql.*;
```

Provides interfaces and classes such as Connection, Statement, and ResultSet.

Required for all JDBC programs.

Register the Driver

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

- Loads and registers the database driver.
- This step tells Java which database driver to use

Establish a Connection

```
Connection con =
DriverManager.getConnection(
  "jdbc:mysql://localhost:3306/testdb",
"root", "password");
```

Connects Java to the database using the correct URL, username, and password.

Returns a Connection object used for executing SQL statements.

Create and Execute Statement

```
Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT *
FROM students");
```

Statement is used to send SQL queries to the database.

Use executeUpdate() for INSERT, UPDATE, DELETE statements

Process Results

```
while(rs.next()) {
  System.out.println(rs.getInt(1) + " " +
rs.getString(2)); }
```

ResultSet is used to read data returned from the database.

The next() method moves the cursor to the next record.

Close the Connection

```
rs.close();

stmt.close();

con.close();
```

Always close all database resources after use.

Prevents memory leaks and connection issues.

Full Example Program

```
import java.sql.*;


public class JDBCExample {
  public static void main(String[] args) {
    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/testdb", "root",
"password");
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery("SELECT * FROM
students");
```

```
 while(rs.next())
        System.out.println(rs.getInt(1) + " " +
rs.getString(2));


rs.close();
      stmt.close();
      con.close();
    } catch(Exception e) {
      System.out.println(e);
    }
  }
}
```

Summary

- JDBC allows Java programs to interact with relational databases.
- JDBC interacts with databases through SQL statements.
- Key terms include Driver, Connection, Statement, and ResultSet.
- It provides portability and consistency for database operations in Java.