Basic J DBC Programming

Presented by:
Shashidhar kadiri

Overview

- Introduction to database.
- JDBC Introduction
  - Drivers
  - Set of classes and Interfaces
  - Steps to perform jdbc Application
- Example
- Few latest drivers
Introduction to Database

- Why do we need Database?
  Database came to existence to replace files which were used to store data.
  The disadvantages of using files are like:
    - Redundancy
    - Inconsistency
    - Security
    - Loss of integrity
    - Poor performance

Database Management System

- **Query Processor**: Checks for syntactical and semantics of query.
- **Database Manager**: Checks for the existence of required table.
- **File Manager**: Directly contacts d/b and retrieves data. It has all the read and write permissions on the d/b.
- Front end like VB with backend Oracle requires a mediator like ODBC (Open Database Connectivity).
Jdbc Introduction

- Jdbc is basically used because it can access any database.
- Jdbc uses drivers to connect to the database, which are categorized into four types.
- Jdbc has a set of exclusive interfaces and classes.
- Few steps to perform Jdbc application.

Jdbc driver (1)

- Jdbc-Odbc bridge driver:
  This bridge driver can be used to connect to any existing database, i.e., ODBC compliant.
JDBC Driver (2)

- **Native-API/partly Java driver**: Converts JDBC calls into database specific calls for database such as SQL, Oracle etc. Driver is partly written in “Java” and partly in “C”.

JDBC Driver (3)

- **Net-protocol/all-Java driver**: This driver is used in Internet to connect to the database at different locations. This is supposed to be the most flexible driver.
**JDBC Driver (4)**

- **Native-protocol/all-Java driver**: Converts the JDBC calls into the network protocol used by DBMS’s directly. This is very much used in intranet access.

**JDBC Architecture.**

```
Application
    | Jdbc Driver Manager
    |   Jdbc-odbc bridge driver
    |       Native-API/partly Java driver
    |           Net-protocol/all-Java driver
    |               Native-protocol/all-Java driver
    |                   Odbc
    |                       Data Base
```

"Odbc"
Getting ready for programming

Used in Jdbc

- **Interfaces**:  
  - Connection  
  - Statement  
  - Result set  
  - Prepared statement  
  - Result set Meta data  
  - Callable statement  
  - D/B Meta data

- **Classes**:  
  - Driver manager  
  - Date Time  
  - Types  
  - Date  
  - Driver property information  
  - Time stamp
Interfaces

- **Connection**: This Interface is used to establish the connection with the database.
- **Statement**: Acts as a handle to take user defined query to the d/b.
- **Result set**: Returns the result of the Query from the database.

Interfaces Contd..

- **Prepared statement**: Takes values at runtime from the user and pass the query to the database.
- **Result set Meta data**: Informs about the result set.
- **Callable Statement**: This statement is used to call the stored procedure.
- **Data Base Meta Data**: Informs about the data itself.
## Classes

- **Driver manager**: Has the list of all drivers present in the system.
- **Types**: Has the constant integer value which are of Sql type.
- **Time stamp**: This gives the latest modified or access time.

## Steps to write Jdbc Application

- Load the driver into memory.
- Establish the connection with the d/b.
- Execute the query.
- Obtain the result set.
- Interpret the result set.
- Close the connection.
Load the driver:

Class forname
("Sun.jdbc.odbc.JdbcOdbcDriver");
This statement is used to load and register the driver in to the memory.

Establish the connection:

Connection
con=DriverManager.getConnection
("Jdbc:Odbc:DSN");
This statement is used to establish the connection with the database.
Whenever getConnection is called in turn it calls 4 methods of the driver interface.
Methods of driver:

- GetDriver(): It gets the first encountered driver which is present in jdbc-odbc driver class.
- Accepts URL(): Checks if the DSN entered is same as the driver loaded.
- Register driver(): Checks if the D/B is eligible to the DSN.
- Connect(): Returns connection object.

Statement:

- Statement st = con.createStatement();
  This allows to assigning the “statement” to a variable “st” which is returned from “CreateStatement”.
**Resultset:**
- Resultset \( rs = \)
  ```java
  st.executeQuery("select * from emp");
  ``
This statement gets the result set in return from the query executed. "rs" has the result set values.

**Interpret the result set:**
- \( \text{int empno=}rs.getInt(1); \)
  As we know "rs" is the result set variable it has an extension of "getInt" which gets the value in that particular column (1). The data type should be correct and known so that the returned value is easily collected.
Close the Connection:

Resultset.close();

This statement closes the connection with the database.

Example

import java.sql.*;
public class first {
    public static void main() {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection con = DriverManager.getConnection("jdbc:odbc:DSN","User","password");
            Statement st = con.createStatement();
            Resultset rs = st.executeQuery("Select * from table");
            While(rs.next()) {
                int ac = rs.getInt(1);
                String name = rs.getString(2);
                String addr = rs.getString(3);
                Float bal = rs.getFloat(4);
            } // while
            System.out.println("Account "+ac);
            System.out.println("Name "+name);
            System.out.println("Address "+addr);
            System.out.println("Balance "+bal);
        } // try
        catch(Exception e) {
            // catch
            rs.close();
        } // main
    } // class
Few latest jdbc drivers

Cloudscape
J turbo
inet Merlia
inet Gate
inet opta

Sources :
- www.Java.sun.com
- www.inetsoftware.de
- www.cloudscape.com