Object Oriented Programming using Java 14

Java Database Connectivity using MySQL

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Steps in Java Application Development

- JDBC APIs are used by a java application to communicate with a database. In otherwords, we use JDBC connectivity code in java application to communicate with a database
- Steps to connect any java application with the database in java using JDBC:
  - Step1: Register the driver class
  - Step2: Creating connection
  - Step3: Creating statement
  - Step4: Executing SQL statements
  - Step5: Closing connection

Steps in Java Application Development...

Steps in Java Application Development...

- **Step1: (Register the driver class)**
  - Register the driver class with driver manager by using *forName()* method of *Class* class
  - Syntax: Class.forName(Driver Classname)
  - Ex: Class.forName(”com.mysql.cj.jdbc.Driver”);
- **Step2: (Creating connection)**
  - Create a connection with database server by using *getConnection()* method of *DriverManager* class
  - Syntax: getConnection(String url, String name, String pwd)
  - Ex: Connection con = DriverManager.getConnection(”jdbc:mysql://localhost:3306/sakila”, ”root”, ”system”);
- **Step3: Creating statement**
  - *createStatement()* method of Connection interface is used to create statement. This statement object is responsible to execute SQL statements with the database
  - Syntax: createStatement()
  - Ex: Statement stmt = con.createStatement();

Steps in Java Application Development...

- **Step4: (Executing SQL statements)**
  - Execute the SQL statements by using *execute()* or *executeUpdate()* or *executeQuery()* method of statement interface
  - *execute()* method is used for CREATE or DROP statement
  - *executeQuery()* method is only used to execute SELECT statement. *executeUpdate()* method is used to execute all SQL statements except SELECT statement
  - Syntax: executeQuery(String query) executeUpdate(String query)
  - Ex: String query = ”Select * from emp”; Resultset rs=stmt.executeQuery(query);

    String query=”insert into emp values(507,’Asis’, 30)”; stmt.executeUpdate(query)
- **Step5: (Closing the connection)**
  - *close()* method of Connection interface is used to close the connection
  - Syntax: close()
  - Ex: con.close();

Connecting with MySQL Database

- For connecting java application with the MySQL database, the following information are required:
  - **Driver class**:"com.mysql.cj.jdbc.Driver"
  - **Connection URL**: "jdbc:mysql://localhost:3306/sakila" where, *3306* is the port number and *sakila* is the MySQL database name
  - **username**: user name for MySQL database
  - **password**: password of the MySQL database user
- To connect java application with MySQL database, *jdbc jar* file is required to be loaded

Statement and PreparedStatement

Statement and PreparedStatement

- Statement is used when you want to run SQL query once
  Statement st = con.createStatement();
  st.executeUpdate("UPDATE STUD SET Name="Ram" WHERE Roll=101");
- PreparedStatement is used when you want to use SQL statements many times. The PreparedStatement interface accepts input parameters at runtime
  PreparedStatement st = con.prepareStatemt("UPDATE STUD SET Name=? WHERE Roll=?");
  st.setString(1,"Ram");
  st.setInt(2, 101);
  st.executeUpdate();
- Statement is used to execute normal SQL queries; whereas PreparedStatement is used to execute dynamic SQL queries
- Parameters can't be passed at runtime in Statement; whereas parameters can be passed at runtime in PreparedStatement

Inserting Record through JDBC

Inserting Record through JDBC

```
import java.sql.*;
public class InsertRecord{
    public static void main(String args[]) throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/sakila", "root", "system");
        Statement stat = con.createStatement();
        stat.executeUpdate("INSERT INTO sakila.emp VALUES (4,'D', 5000)");
        System.out.println("Records inserted into table successfully...");
    }
}
```

Selecting/Querying Record through JDBC

- ResultSet is essentially a table of data where each row represents a record and each column represents a field in database. It has a cursor that points to the current row in the ResultSet and we can able to navigate in ResultSet by using the next(), previous(), first(), and last() methods
- Data can be retrieved by using different methods like getString(), getInt(), getDouble() etc.

```
import java.sql.*;
public class SelectRecord{
    public static void main(String args[]) throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/sakila", "root", "system");
        Statement stat = con.createStatement();
        ResultSet rs = stat.executeQuery("Select * from emp");
        while (rs.next()){
                System.out.println(rs.getInt(1)+" "+rs.getString(2)+
                " "+rs.getDouble(3));
            }
        System.out.println("Records retrieved successfully...");
    }
}
```

Updating Record through JDBC

Updating Record through JDBC

```
import java.sql.*;
public class UpdateRecord{
    public static void main(String args[]) throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/sakila", "root", "system");
        Statement stat = con.createStatement();
        stat.executeUpdate("Update emp set Sal=2500 where eid=2");
        System.out.println("Records updated successfully...");
    }
}
```

Deleting Record through JDBC

Deleting Record through JDBC

```
import java.sql.*;
public class DeleteRecord{
    public static void main(String args[]) throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/sakila", "root", "system");
        Statement stat = con.createStatement();
        stat.executeUpdate("Delete from emp where eid=3");
        System.out.println("Records deleted successfully...");
    }
}
```

Parameterized Query /Prepared Statement

Parameterized Query /Prepared Statement

```
import java.sql.*;
public class ParameterizedQuery{
    public static void main(String args[]) throws Exception{
        ResultSet result;
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/sakila", "root", "system");
        PreparedStatement stat1 = con.prepareStatement("insert into emp values(?,?,?)");
        stat1.setInt(1,Integer.parseInt(args[0]));
        stat1.setString(2,args[1]);
        stat1.setDouble(3,Double.parseDouble(args[2]));
        stat1.executeUpdate();
        System.out.println("Records inserted and displayed successfully");
        Statement stat2 = con.createStatement();
        ResultSet rs = stat2.executeQuery("Select * from emp");
        while (rs.next()){
                System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getDouble(3));
            }
    }
}
```