JDBC PART-2

Type - IV Driver (JDBC 100% Pure Java Driver):

It is also called as JDBC Native Protocol Driver (or) Thin Driver

Type - IV Driver Class Name for Oracle Database:

oracle.jdbc.driver.OracleDriver

URL to access driver:

jdbc:oracle:thin:@domain-name:port-no:service-id

Type - IV Driver Functionality:

It passes the java instructions directly to a database.

Advantages:

1) It is a highest performance driver as compared to all other drivers.

2) DSN not required.

3) Database not needed on same system.

4) It is suitable for applets.

Disadvantages:

1) Separate driver required for every database.

Program to establish a connection between Java application & oracle database by using type-iv driver:

```
import java.sql.*;
class ConnectionDemo
{
        public static void main(String args[])
        {
```

```
            try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");
            System.out.println("Connection Established Successfully");
            }catch(Exception e)
            {
                  System.err.println(e);
            }
      }
}
```

Methods of Connection interface:

1) public abstract java.sql.Statement createStatement() throws java.sql.SQLException;

2) public abstract java.sql.PreparedStatement prepareStatement(java.lang.String) throws java.sql.SQLException;

3) public abstract java.sql.CallableStatement prepareCall(java.lang.String) throws java.sql.SQLException;

Statement interface:

It is used to execute static SQL queries.

PreparedStatement interface:

It is used to execute dynamic SQL queries.

CallableStatement interface:

It is used to execute PL/SQL programs.

Methods of Statement interface:

1) public abstract boolean execute(java.lang.String) throws java.sql.SQLException;

2) public abstract int executeUpdate(java.lang.String) throws java.sql.SQLException;

3) public abstract java.sql.ResultSet executeQuery(java.lang.String) throws java.sql.SQLException;

execute() method:

It is suitable to execute DDL queries. DDL stands for Data Definition Language.

Examples: CREATE, ALTER, DROP, .. etc.,

executeUpdate() method:

It is suitable to execute DML queries. DML stands for Data Manipulation Language.

Examples: INSERT, UPDATE, DELETE, .. etc.,

executeQuery() method:

It is suitable to execute DQL queries. DQL stands for Data Query Language.

Examples: SELECT

Program to create a table:

```
import java.sql.*;
class CreateDemo
{
	public static void main(String args[])
	{
		try{
		Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
            Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");

            Statement stmt=con.createStatement();

            stmt.execute("create table student(rollno number(3), name
varchar2(10), marks number(3))");

            System.out.println("Table Created Successfully");

            }catch(Exception e)

            {

                  System.err.println(e);

            }

      }

}
```

Program to insert a record:

```
import java.sql.*;

class InsertDemo

{

      public static void main(String args[])

      {

            try{

            Class.forName("oracle.jdbc.driver.OracleDriver");

            Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");

            Statement stmt=con.createStatement();

            stmt.executeUpdate("insert into student values(1, 'Venkatesh', 88)");
```

```
            System.out.println("One Record Inserted Successfully");
            }catch(Exception e)
            {
                  System.err.println(e);
            }
      }
}
```

Program to retrieve data from database:

```
import java.sql.*;
class SelectDemo
{
      public static void main(String args[])
      {
            try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","syste
m","manager");
            Statement stmt=con.createStatement();
            ResultSet rs=stmt.executeQuery("select * from student");
            ResultSetMetaData rm=rs.getMetaData();
            int n=rm.getColumnCount();
            for(int i=1;i<=n;i++)
            {
                  System.out.print(rm.getColumnName(i)+"\t");
```

```
            }
            System.out.println();
            while(rs.next())
            {
                  for(int i=1;i<=n;i++)
                  {
                        System.out.print(rs.getString(i)+"\t");
                  }
                  System.out.println();
            }
            }catch(Exception e)
            {
                  System.err.println(e);
            }
      }
}
```


By

Mr. Venkatesh Mansani

Naresh i Technologies