"शीलं परं भूषणम्"

M. K. S. S. S.

CUMMINS COLLEGE OF ENGINEERING FOR WOMEN, PUNE.

STUDENT'S ROLL NO. : 3330

Assignment No.-6

Implement MySQL Java Connectivity (2 tier)

1. What is a 2 tier application?

JDBC architecture consists of 2 tier & 3 tier processing models to access a database. Two tier model- A java application communicates directly with to the data source. The JDBC driver enables the communication between the application and the data source. When a user sends a query to the data source, the answers to those queries are sent back to the user in the form of results. The data source can be located on a different machine, on a network to which the user is ~~treated~~ connected. This is known as the client/server configuration, where the user's machine acts a client & the machine having the data source running acts as the server.

2. What is JDBC?

JDBC stands for Java Database Connectivity, which is a standard API for database independent connectivity between the java programming language and a wide range of databases.

The JDBC library includes API's for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database
- Creating MySQL statements
- Executing MySQL queries in the database.

- Viewing & modifying the resulting records.

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allow for portable access to an underlying database. Java can be used to write different types of executables such as -

- Java applications
- java applets
- java servlets
- java server pages (JSPs)
- enterprise java beans (EJBs)

All of these different executables are able to use a JDBC driver to access a database & take advantage of the stored data.

- JDBC provides the same capabilities as ODBC, allowing java programs to contain database independent code.

3. What is a driver? Explain the different types of drivers.

JDBC drivers are Client Side Adapters (installed on the Client machine, not on the Server) that convert the requests from the java programs to a protocol that the DBMS can understand. There are 4 types of JDBC drivers -

1) Type 1 driver or JDBC-ODBC bridge driver.

2) Type 2 driver or Native API driver.

3) Type 3 driver or Network Protocol Driver.

4) Type 4 driver or Thin driver.

Type-1 driver -

Type-1 driver or JDBC-ODBC driver uses ODBC driver to connect to the database. The JDBC-ODBC bridge driver converts JDBC method calls into the ODBC function calls. Type-1 driver is also called as Universal driver because it can be used to connect to any of the databases.

As a common driver is used to interact with different databases, the data transferred through the driver is not so secured.

The ODBC driver is needed to be installed in individual client machines.

Type 1 driver isn't written in java, that's why it isn't a portable driver. This driver software is built in with SDK so no need to install separately.

It is a database-independent driver.

Type 2 - driver -

The Native API driver uses the Client side libraries of the database. This driver converts the JDBC method calls into the native calls of the database APIs. In order to interact with different database, this driver needs their local API, hence the data transfer is much more secure as compared to the type1

driver. Driver needs to be installed separately on individual client machines. Type 2 driver is not written in java hence it is not a portable driver. It is a database dependent driver.

Type 3 driver - The Network Protocol driver uses middleware (application server) that converts JDBC calls directly or indirectly into ~~vect~~ vendor specific database protocol. Here all the database connectivity drivers are present in a single server, hence no need of individual client side installation. Type 3 drivers are fully written in java, hence they are portable drivers. No client side library is required because of the application server that can perform many tasks like auditing, load balancing, logging, etc. There is switch facility to switch from one database to another database.

~~4. What is a jar file? Write the steps to link JAR file.~~

Type 4 driver - Also called as Native Protocol driver. The driver interacts directly with the database. It does not require any native database library hence it is also called as thin driver. Does not require any native library and middle ware server so no client side or server side installation. It is fully written in java language hence they are portable drivers.

Which driver to use when? → If you are using one type of a database then the preferred driver type is type 4.

If your java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.
Type 2 drivers are suitable in situations where a type 3 or a type 4 driver is not available yet for your database.
Type 1 driver is not considered as a deployment level driver and is used for development & testing purposes only.

4. What is a JAR file? What are the steps to link a JAR file.
JAR is an abbreviation of JAVA archive. It is used for aggregating multiple files into a single one, and it is present in a ZIP format.
JAR file can be added in a classpath in the following ways -
1. Using Eclipse / any IDE
2. Using command line

Method 1 - Using Eclipse IDE
Step 1 - Right Click on your project name
Step 2 - Click on Build Path
Step 3 - Click on configure build path
Step 4 - Click on libraries and click on "Add External JARs".
Step 5 - Select the jar file from the folder where you have saved the JAR file.
Step 6 - Click on Apply & OK.

Method 2 - Using the Command line.

Command 1 - By including JAR name in classpath environment variable

Command 2 - By including name of JAR file in a classpath command line option.

Command 3 - By including the jar name in the class-Path option in the manifest.

Command 4 - By using Java 6 wildcard option to include multiple JAR.

5. What is the purpose of class.forName() method?

The forName() method of java.lang.Class class is used to get the instance of this class with the specified class name. This class name is specified as the string parameter.

public static Class<T> forName (String className) throws ClassNotFoundException

Parameter - This method accepts the parameter className which is the class for which its instance is required.

Return value - This method returns the instance of this class with the specified class name.

Exception - This method throws the following exceptions -

LinkageError - If the linkage fails.

ExceptionInitializeError - If the initialization provoked by this method fails.

ClassNotFoundException - If the class cannot be located.

6. What is a ResultSet?
A ResultSet object maintains a cursor that points to the current row in the result set. The term "result set" refers to the row and column data contained in a Resultset object.
The methods of the ResultSet interface can be broken down into 3 categories -

a. Navigational methods - Used to move the cursor around.

b. Get methods - Used to view the data in the columns of the current row being pointed by the cursor.

c. Update methods - Used to update the data in the columns of the current Row. The updates can then be updated in the underlying database as well.

The cursor is movable based on the properties of the ResultSet. These properties are designated when the corresponding statement that generates the resultSet is created. JDBC provides the following connection methods to create statements with desired ResultSet -

→ createStatement(int RS_type, int RSConcurrency);

→ prepareStatement(String SQL, int RSType, int RSConcurrency);

~~8. Explain Statement and Prepared statement concept.~~

→ PrepareCall(String SQL, int RSType, int RS Concurrency);

The first argument indicates the type of a ResultSet object. The 2nd argument is one of the 2 ResultSet constants for specifying whether a result set is read only or updatable.

7. Explain Statement and Prepared statement concept.

Statement - It is used when SQL query is to be executed only once. It is used for general (create, alter, drop statements) purpose access to your database. Statement is useful when you are using static SQL statements at runtime. Statement interface cannot accept parameters. Performance is low for statement.

example - Statement is base interface.

```
// Creating the Statement Object
↓
statement GFG = con.createStatement();

//Executing the Statement

GFG.executeUpdate("CREATE TABLE STUDENT
                    (ID NUMBER NOT NULL, NAME VARCHAR)");
```

Prepared Statement - It is used when you want to use SQL statements many times. You can pass parameters at runtime. It is used for queries which are to be executed multiple times. The performance for Prepared Statement is better than Statement. It extends the statement interface. Prepared Statement is used for executing dynamic SQL ~~querries.~~ queries.
Unlike statement, we can use prepared statement for reading and writing binary data.