Bootcamp Training

Java Spring AWS Frontend Training

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

Authorized & published by Summitworks Technologies Inc

Agenda: Day -5

- Java Database Connectivity (JDBC)
  - The steps in implementing a JDBC application
  - The JDBC mechanism-Connecting to a DB
  - Types of statements, Result Sets etc.
  - Statement & Prepared Statement Examples
- Introduction to ORM and Hibernate
- Real-time Examples

Introduction to JDBC

JDBC stands for "Java Database Connectivity". It is an API (Application Programming Interface) which consists of a set of Java classes, interfaces and exceptions and a specification to which both **JDBC driver vendors** and **JDBC developers** (like we) adhere when developing applications.

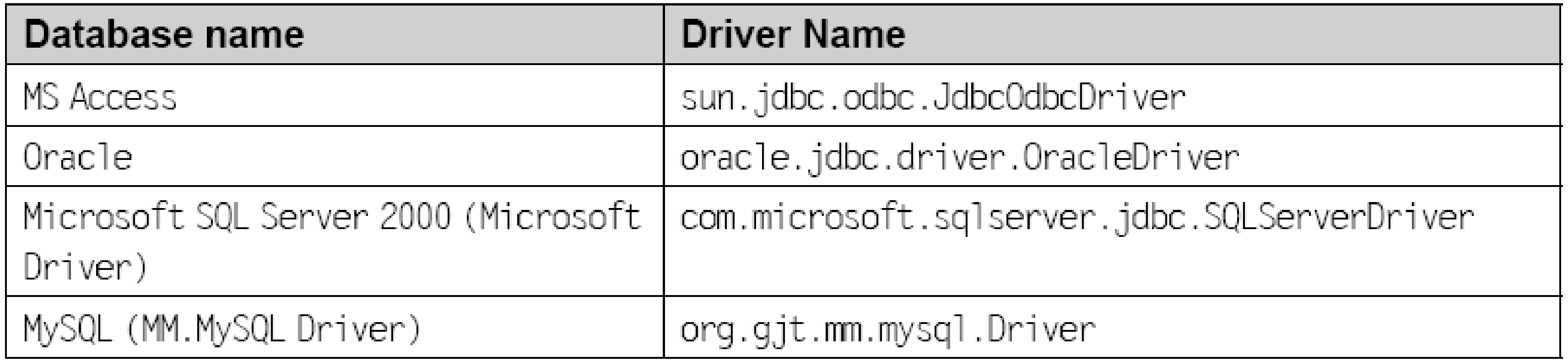
Database handling using JDBC

- JDBC stands for Java database connectivity.
- a standard API for all Java programs to connect to databases. The JDBC API is available in two packages:
  - Core API *java.sql*.
  - Standard extension to JDBC API *javax.sql* (supports connection pooling, transactions, etc.)
- JDBC defines a few steps to connect to a database and retrieve/insert/update databases.
- The steps are as follows:
  - Load the driver
  - Establish connection
  - Create statements
  - Execute query and obtain result
  - Iterate through the results

Load the Driver

- The driver is loaded with the help of a static method,
  - Class.forName(drivername)
- Every database has its own driver.

Driver Names

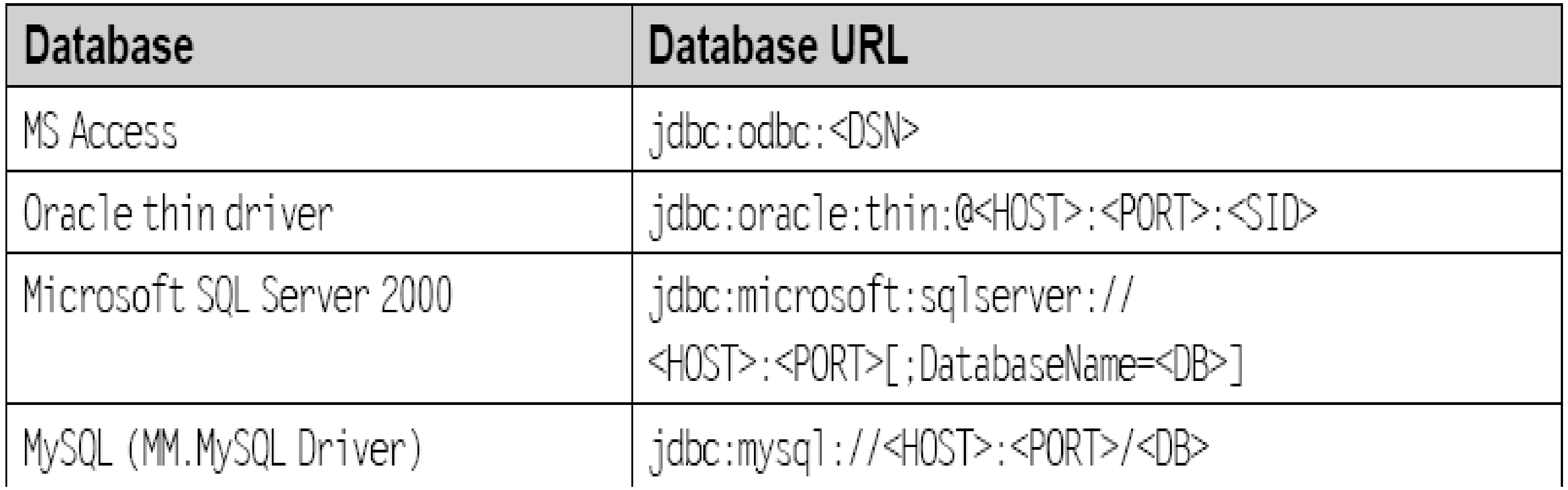


Database name	Driver Name
MS Access	sun.jdbc.odbc.JdbcOdbcDriver
Oracle	oracle.jdbc.driver.OracleDriver
Microsoft SQL Server 2000 (Microsoft Driver)	com.microsoft.sqlserver.jdbc.SQLServerDriver
MySQL (MM.MySQL Driver)	org.gjt.mm.mysql.Driver

Establish a Connection

- A connection to the database is established using the static method *getConnection(databaseUrl)* of the DriverManager class.
- The DriverManager class is class for managing JDBC drivers.
- The database URL takes the following shape *jdbc:subprotocol:subname*.
- If any problem occurs during accessing the database, an SQLException is generated, else a Connection object is returned which refers to a connection to a database.
- Connection is actually an interface in *java.sql* package.
  - Connection con=DriverManager.getConnection(databaseUrl);

Few Database URLs



Database	Database URL
MS Access	jdbc:odbc:<DSN>
Oracle thin driver	jdbc:oracle:thin:@<HOST>:<PORT>:<SID>
Microsoft SQL Server 2000	jdbc:microsoft:sqlserver://<HOST>:<PORT>[;DatabaseName=<DB>]
MySQL (MM.MySQL Driver)	jdbc:mysql://<HOST>:<PORT>/<DB>

Create Statement

- The connection is used to send SQL statements to the database.
- three interfaces are used for sending SQL statements to databases
  - Statement and its two sub-interfaces,
  - PreparedStatement and Callable Statement.
- Three methods of the Connection object are used to return objects of these three statements.
- A Statement object is used to send a simple SQL statement to the database with no parameters.
  - Statement stmt = con.createStatement();

Create Statement (contd.)

- A PreparedStatement object sends precompiled statements to the databases with or without IN parameters.
- If n rows need to be inserted, then the same statement gets compiled n number of times.
- So to increase efficiency, we use precompiled PreparedStatement.
- only the values that have to be inserted are sent to the database again and again.
  - PreparedStatement ps = con.prepareStatement(String query);
- A CallableStatement object is used to call stored procedures.
  - CallableStatement cs = con.prepareCall(String query);

Execute Query

- Three methods are used
  - ResultSet executeQuery(String sqlQuery) throws SQLException
  - int executeUpdate(String sqlQuery) throws SQLException
- executeQuery is used for executing SQL statements that return a single ResultSet, e.g. a select statement.
  - The rows fetched from database are returned as a single ResultSet object. For example,
  - ResultSet rs=stmt.executeQuery(“select * from emp”);
- executeUpdate is used for DDL and DML SQL statements like insert,update, delete, and create.
  - returns an integer value for DML to indicate the number of rows affected and 0 for DDL statements which do not return anything.

Execute Query (contd.)

  - PreparedStatement ps = con.prepareStatement(“update emp set salary=? where empid=?”);
- The statement is sent to database and is prepared for execution, only the value of the IN (?) parameters need to be sent.
  - ps.setInt(1,100000);
  - ps.setString(2,”Emp001”);
  - ps.executeUpdate();
- The execute method is used when the statement may return more than one ResultSet or update counts or a combination of both.
- This happens when stored procedures are executed.

Iterate ResultSet

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

Result Set Meta Data

```
ResultSetMetaData rsmd=rs.getMetaData();
    System.out.println("Column in ResultSet:"+rsmd.getColumnCount());
    for(int i=1;i<=rsmd.getColumnCount();i++)
    {
    System.out.println("Column Name :"+rsmd.getColumnName(i));
    System.out.println("Column Type :"+rsmd.getColumnTypeName (i));
    }
```

Example

```
import java.sql.*;
    class DatabaseConnection{
    public static void main(String args[]) throws Exception{
    Class.forName(“sun.jdbc.odbc.JdbcOdbcDriver”);
    Connection con=DriverManager.getConnection(“jdbc:odbc:sac”);
    PreparedStatement ps=con.prepareStatement(“insert into emp values (?,?,?)”);
    ps.setString(1,”Emp001”);
    ps.setString(2,”Peter”);
    ps.setInt(3,10000);
    System.out.println(“Row inserted : “+ps.execute Update());
    Statement stmt=con.createStatement();
    ResultSet rs=stmt.executeQuery(“select * from emp”);
```

Example (contd.)

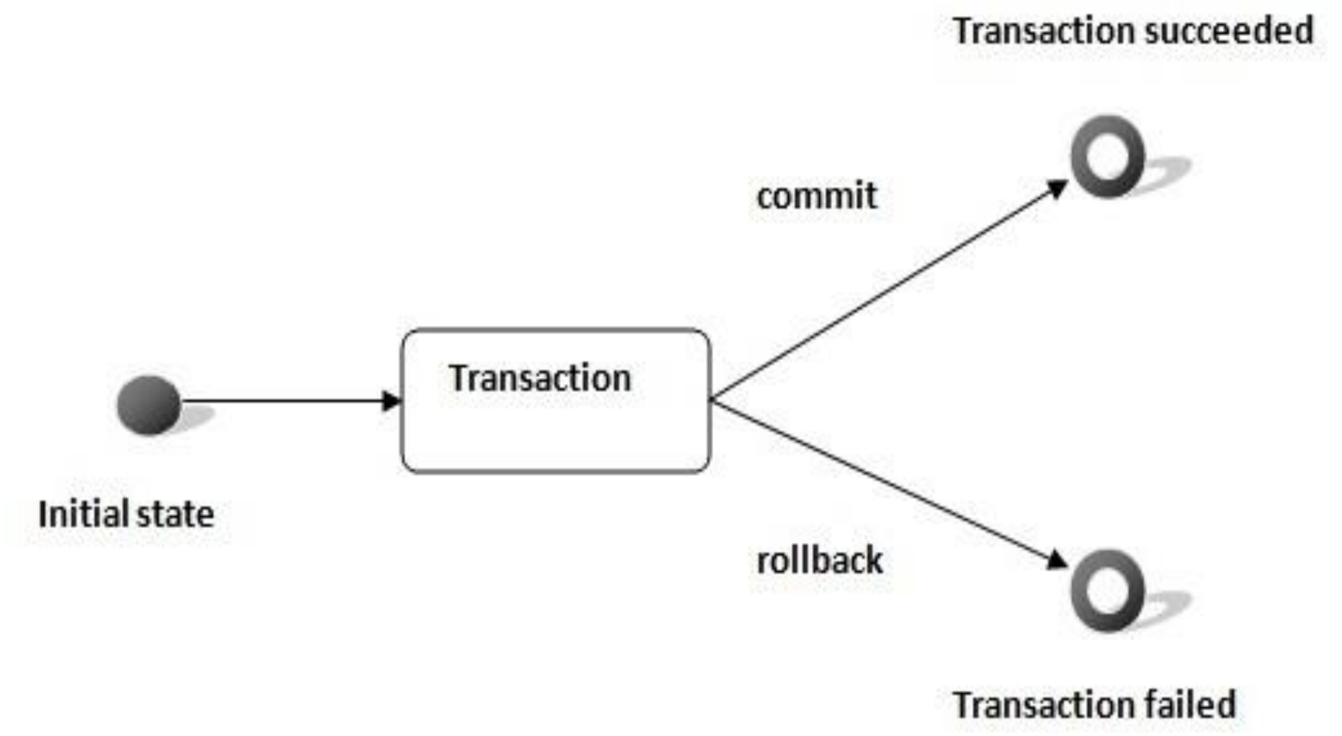
```
ResultSetMetaData rsmd=rs.getMetaData();
    int cc=rsmd.getColumnCount();
    System.out.println("Number of columns in result set: "+cc);
    for(int i=1;i<=cc;i++)
    System.out.print(rsmd.getColumnName(i)+"\t");
    System.out.println();
    while(rs.next()){
    System.out.print(rs.getString(1)+"\t");
    System.out.print(rs.getString(2)+"\t");
    System.out.print(rs.getString(3)+"\n");} } }
```

Transaction Management in JDBC

- Transaction represents **a single unit of work**.
- The ACID properties describes the transaction management well. ACID stands for Atomicity, Consistency, isolation and durability.
- **Atomicity** means either all successful or none.
- **Consistency** ensures bringing the database from one consistent state to another consistent state.
- **Isolation** ensures that transaction is isolated from other transaction.
- **Durability** means once a transaction has been committed, it will remain so, even in the event of errors, power loss etc.

Advantage of Transaction Management

- **fast performance** It makes the performance fast because database is hit at the time of commit.



Transaction Management

- In JDBC, **Connection interface** provides methods to manage transaction

Method	**Description**
void setAutoCommit(boolean status)	It is true by default means each transaction is committed by default.
void commit()	commits the transaction.
void rollback()	cancels the transaction.

Batch Processing in JDBC

- Instead of executing a single query, we can execute a batch (group) of queries. It makes the performance fast.
- The java.sql.Statement and java.sql.PreparedStatement interfaces provide methods for batch processing.
- **Advantage of Batch Processing**
- Fast Performance

Example of batch processing using PreparedStatement

- **import** java.sql.*;
- **class** FetchRecords{
- **public static void** main(String args[])**throws** Exception{
- Class.forName("oracle.jdbc.driver.OracleDriver");
- Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system",“xe");
- con.setAutoCommit(**false**);
- Statement stmt=con.createStatement();
- stmt.addBatch("insert into user420 values(190,'abhi',40000)");
- stmt.addBatch("insert into user420 values(191,'umesh',50000)");
- stmt.executeBatch();//executing the batch
- con.commit();
- con.close();
- }}

Example to store image in Oracle database

You can store images in the database in java by the help of **PreparedStatement** interface.

The **setBinaryStream()** method of PreparedStatement is used to set Binary information into the parameterIndex.

The syntax of setBinaryStream() method is given below:

1) **public void** setBinaryStream(**int** paramIndex,InputStream stream)**throws** SQLException.

```
Query: CREATE TABLE  "IMGTABLE"
   (    "NAME" VARCHAR2(4000),
    "PHOTO" BLOB
 )
```

Example to retrieve image from Oracle database

- By the help of **PreparedStatement** we can retrieve and store the image in the database.
- The **getBlob()** method of PreparedStatement is used to get Binary information, it returns the instance of Blob. After calling the **getBytes()** method on the blob object, we can get the array of binary information that can be written into the image file.
- Signature of getBlob() method of PreparedStatement

public Blob getBlob()**throws** SQLException

- Signature of getBytes() method of Blob interface

public **byte**[] getBytes(**long** pos, **int** length)**throws** SQLException

Java CallableStatement Interface

- CallableStatement interface is used to call the **stored procedures and functions**.
- We can have business logic on the database by the use of stored procedures and functions that will make the performance better because these are precompiled.
- Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.

Oracle Example

```
CREATE OR REPLACE PROCEDURE getEmpName
   (EMP_ID IN NUMBER, EMP_FIRST OUT VARCHAR) AS
BEGIN
   SELECT name INTO EMP_FIRST
  FROM Employees
   WHERE ID = EMP_ID;
END;
```

Mysql

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `EMP`.`getEmpName` $$
CREATE PROCEDURE `EMP`.`getEmpName`
   (IN EMP_ID INT, OUT EMP_FIRST VARCHAR(255))
BEGIN
   SELECT first INTO EMP_FIRST
   FROM Employees
   WHERE ID = EMP_ID;
END $$

DELIMITER ;
```

Sample function code

- create or replace function sum4
- (n1 in number,n2 in number)
- **return** number
- is
- temp number(8);
- begin
- temp :=n1+n2;
- **return** temp;
- end;
- /

Queries?