Introduction to JDBC

IT Workshop II

CS 251

JDBC

- JDBC (Java Database Connectivity) API allows Java programs to connect to databases

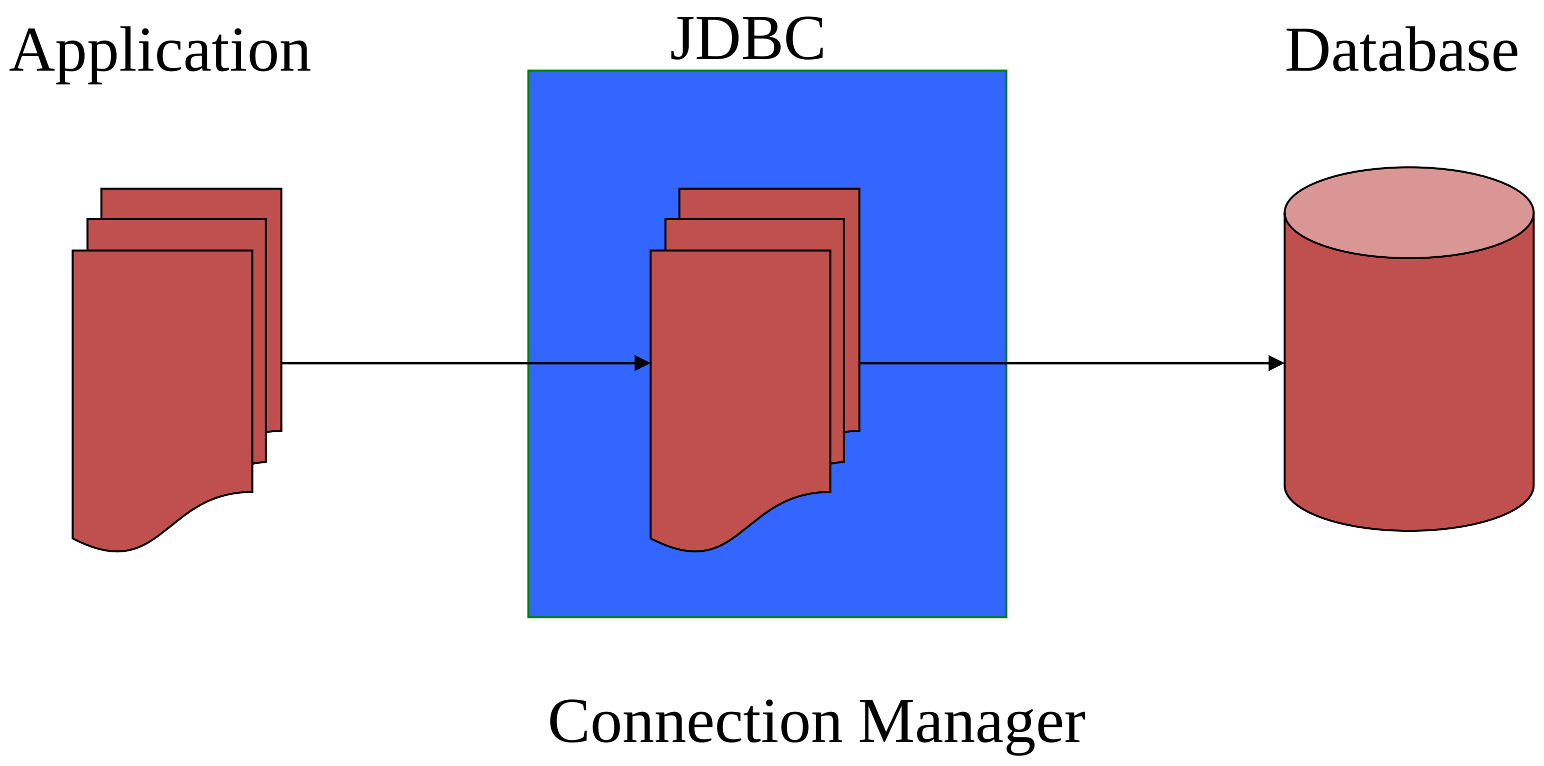
- The JVM uses a JDBC driver to translate generalized JDBC calls into vendor specific database calls

- There are four general types of JDBC drivers
  – We will look at Type 4 …

Pure Java Driver (Type 4)

- These drivers convert the JDBC API calls to direct network calls by making direct socket connections with the database
- It is the most efficient method to access database, both in performance and development time
- It is the simplest to deploy
- All major database vendors provide pure Java JDBC drivers for their databases and they are also available from third party vendors

JDBC Data Source Architecture



Driver Path - Command Line Program and Eclipse

- Find the driver jar file

  (named *mysql-connector-java-5.1.39-bin.jar)*
- Set an environment variable named *CLASSPATH* with the path of the jar file.

- For *Eclipse,* just copy the jar file into the project.

Typical JDBC Programming Procedure

The steps of applying JDBC are:

1. Load the JDBC driver that is specific to DB
2. Get a Connection object
3. Get a Statement object
4. Execute queries and/or updates
5. Read results
6. Read Meta-data (optional step)
7. Close Statement and Connection objects

Driver Manager

- The purpose of the java.sql.DriverManager class in JDBC is to provide a common access layer on top of different database drivers used in an application

- DriverManager requires that each driver required by the application must be registered before use, so that the DriverManager is aware of it

- Load the database driver using ClassLoader (till jdk 1.5) :

```
Class.forName("com.mysql.jdbc.Driver");
```

Connecting to a Database

- Type 4 JDBC Driver – MySQL Server

```
Connection connect = DriverManager.getConnection (
"jdbc:mysql://url:port_no/database_name?user=user_name&passwo
   rd=your_password&useSSL=false"
);
```

Example:

```
connect = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/MYDB?user=root&password=sanjay&u
   seSSL=false"
);
```

Creating Statement Object

- Creating JDBC statements

```
Statement stmt = connect.createStatement ();
```

- Execute a statement – 3 types:
  - executeUpdate() - for create, insert, update, delete etc.
  - executeQuery() - for getting the data from database
  - execute() - any kind of operations

Execute Statements

- stmt.executeUpdate ("MySQL create table command");
- This uses executeUpdate because the SQL statement contained in createTable is a DDL (data definition language) statement
- Statements that create a table, alter a table, or drop a table are all examples of DDL statements and are executed with the method executeUpdate
- executeUpdate is also used to execute SQL statements that update a table

Execute Statements

- In practice, executeUpdate is used far more often to update tables than it is to create them because a table is created once but may be updated many times

- The method used most often for executing SQL statements is executeQuery

- executeQuery is used to execute SELECT statements, which comprise the vast majority of SQL statements

Entering Data into a Table

```
Statement stmt = con.createStatement();

stmt.executeUpdate (
 “INSERT INTO table_name VALUES (,,,)”
);
```

ResultSet

- executeQuery(String ***sql***) returns a ResultSet
  - ResultSet has a very large number of get*XXX* methods, such as
    - public String getString(String ***columnName***)
    - public String getString(int ***columnIndex***)
  - Results are returned from the current row
  - You can iterate over the rows:
    - public boolean next()
- ResultSet objects, like Statement objects, should be closed when you are done with them
  - public void close()

Getting Data From a Table

```
ResultSet resultSet = null;
resultSet = stmt.executeQuery("select * from STUDENT");

while (resultSet.next()) {
String userID = resultSet.getString("ID");
String userName = resultSet.getString("NAME");
System.out.println("ID : " + userID + " Name : " + userName);
}
```