Object Oriented Programming in Java

Er.Sital Prasad Mandal

BCA- 3th sem
Mechi Multiple Campus
Bhadrapur, Jhapa, Nepal
(Email : info.sitalmandal@gmail.com)
https://ctaljava.blogspot.com/

Text Book

1. Deitel & Dietel. -Java: How to-program-. 9th Edition. TearsorrEducation. 2011, ISBN: 9780273759168
2. Herbert Schildt. "Java: The CoriviaeReferi4.ic e 61 Seventh Edition. McGraw -Hill 2006, 1SBN; 0072263857

Database Programming using JDBC

1. **Using Connection,**
2. **Statement & Result Set Interfaces for Manipulating Data with the Databases.**

What is JDBC?

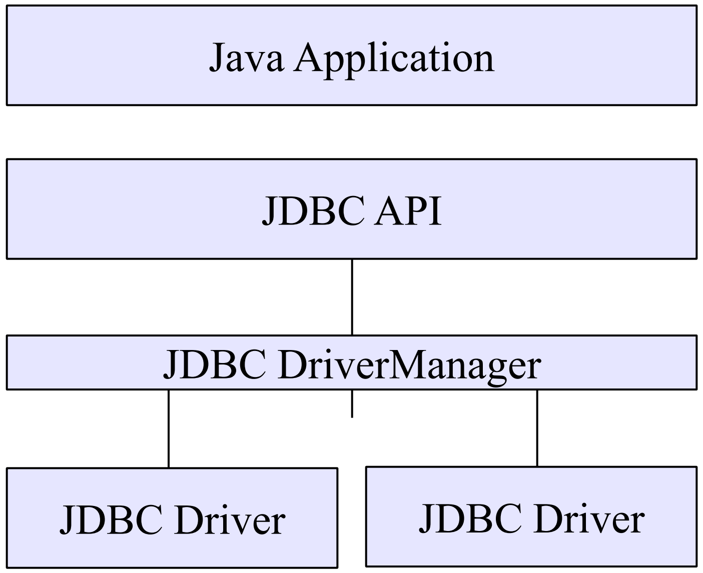
- JDBC provides Java applications with access to most database systems via SQL
- The architecture and API closely resemble Microsoft's ODBC
- JDBC 1.0 was originally introduced into Java 1.1
  - JDBC 2.0 was added to Java 1.2
- JDBC is based on SQL-92
- JDBC classes are contained within the java.sql package
  - There are few classes
  - There are several interfaces

Database Connectivity History

- Before APIs like JDBC and ODBC, database connectivity was Slow:
  - Each database vendor provided a function library for accessing their database.
  - The connectivity library was proprietary.
  - If the database vendor changed for the application, the data access portions had to be rewritten.
  - If the application was poorly structured, rewriting its data access might involve rewriting the majority of the application.
  - The costs incurred generally meant that application developers were stuck with a particular database product for a given application.

JDBC Architecture

- With JDBC, the application programmer uses the JDBC API
  - The developer never uses any proprietary APIs
- Any proprietary APIs are implemented by a JDBC driver
  - There are 4 types of JDBC Drivers



JDBC Drivers

- There are 4 types of JDBC Drivers
  - Type 1 - JDBC-ODBC Bridge
  - Type 2 - JDBC-Native Bridge
  - Type 3 - JDBC-Net Bridge
  - Type 4 - Direct JDBC Driver
- Type 1 only runs on platforms where ODBC is available
  - ODBC must be configured separately
- Type 2 Drivers map between a proprietary Database API and the JDBC API
- Type 3 Drivers are used with middleware products
- Type 4 Drivers are written in Java
  - In most cases, type 4 drivers are preferred

JDBC Classes

- DriverManager
  - Manages JDBC Drivers
  - Used to Obtain a connection to a Database
- Types
  - Defines constants which identify SQL types
- Date
  - Used to Map between java.util.Date and the SQL DATE type
- Time
  - Used to Map between java.util.Date and the SQL TIME type
- TimeStamp
  - Used to Map between java.util.Date and the SQL TIMESTAMP type

JDBC Interfaces

- Driver
  - All JDBC Drivers must implement the Driver interface. Used to obtain a connection to a specific database type

- Connection
  - Represents a connection to a specific database
  - Used for creating statements
  - Used for managing database transactions
  - Used for accessing stored procedures
  - Used for creating callable statements

- Statement
  - Used for executing SQL statements against the database

JDBC Interfaces

- ResultSet
  - Represents the result of an SQL statement
  - Provides methods for navigating through the resulting data
- PreparedStatement
  - Similar to a stored procedure
  - An SQL statement (which can contain parameters) is compiled and stored in the database
- CallableStatement
  - Used for executing stored procedures
- DatabaseMetaData
  - Provides access to a database's system catalogue
- ResultSetMetaData
  - Provides information about the data contained within a ResultSet

Using JDBC

- To execute a statement against a database, the following flow is observed
  - Load the driver (Only performed once)
  - Obtain a Connection to the database (Save for later use)
  - Obtain a Statement object from the Connection
  - Use the Statement object to execute SQL. Updates, inserts and deletes return Boolean. Selects return a ResultSet
  - Navigate ResultSet, using data as required
  - Close ResultSet
  - Close Statement

- Do NOT close the connection
  - The same connection object can be used to create further statements
  - A Connection may only have one active Statement at a time. Do not forget to close the statement when it is no longer needed.
  - Close the connection when you no longer need to access the database

Loading Drivers

- Even a good API can have problems
  - Loading drivers fits into this category
- The DriverManager is a singleton
- Each JDBC Driver is also a singleton
- When a JDBC Driver class is loaded, it must create an instance of itself and register that instance with the JDBC DriverManager
- How does one load a "class" into the Virtual machine?
  - Use the static method Class.forName()

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

13. Database Programming using JDBC

Connecting to a Database

- Once a Driver is loaded, a connection can be made to the database

- The connection is defined by URL
  - The URL has the following form:

    jdbc:*driver:databasename*

  - Examples:

    jdbc:odbc:MyOdbcDatabase

    jdbc:postgres:WebsiteDatabase

    jdbc:oracle:CustomerInfo

- A connection is obtained in the following manner:

```
Connection aConnection = DriverManager.getConnection("jdbc:odbc:myDatabase");
```

- Overloaded versions of the getConnection method allow the specification of a username and password for authentication with the database.

Using a Connection

- The Connection interface defines many methods for managing and using a connection to the database

```
public Statement createStatement()
public PreparedStatement prepareStatement(String sql)
public void setAutoCommit(boolean)
public void commit()
public void rollback()
public void close()
```

- The most commonly used method is createStatement()
  - When an SQL statement is to be issued against the database, a Statement object must be created through the Connection

Using a Statement

- The Statement interface defines two methods for executing SQL against the database

```
public ResultSet executeQuery(String sql)
public int executeUpdate(String sql)
```

- executeQuery returns a ResultSet
  - All rows and columns which match the query are contained within the ResultSet
  - The developer navigates through the ResultSet and uses the data as required.
- executeUpdate returns the number of rows changed by the update statement
  - This is used for insert statements, update statements and delete statements

Using a ResultSet

- The ResultSet interface defines many navigation methods

```
public boolean first()
public boolean last()
public boolean next()
public boolean previous()
```

- The ResultSet interface also defines data access methods

```
public int getInt(int columnNumber)   -- Note: Columns are numbered
public int getInt(String columnName)-- from 1 (not 0)
public long getLong(int columnNumber)
public long getLong(String columnName)
public String getString(int columnNumber)
public String getString(String columnName)
```

- There are MANY more methods. Check the API documentation for a complete list

13. Database Programming using JDBC

Example code

```
public class MySQLconnection {
    public static void main(String[] args)throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/employee","root","");
PreparedStatement statement = con.prepareStatement("select * from emp1");
        ResultSet rs = statement.executeQuery();
        while (rs.next())
        {
            System.out.println("Sucess");
            System.out.println(rs.getString(1)+" "+rs.getString(2));

         }
    con.close();
}
}
```

Motivate

We fall. We fail. We break.
But then, we rise. We heal.
We overcome.
@successpictures