JAC444 - Introduction to Java for C++ Programmers

Lesson 11: JDBC

Agenda

- What is JDBC?
- The JDBC framework
- JDBC Drivers
- JDBC Programming

What is JDBC?

- JDBC is the Java API for accessing relational database.
  - JDBC is the trademark name instead of an acronym.
  - JDBC is often thought to stand for Java Database Connectivity.
- The JDBC API defines classes to represent constructs such as database connections, SQL statements, result sets, and database metadata.
- JDBC allows a Java program to issue SQL statements and process the results.

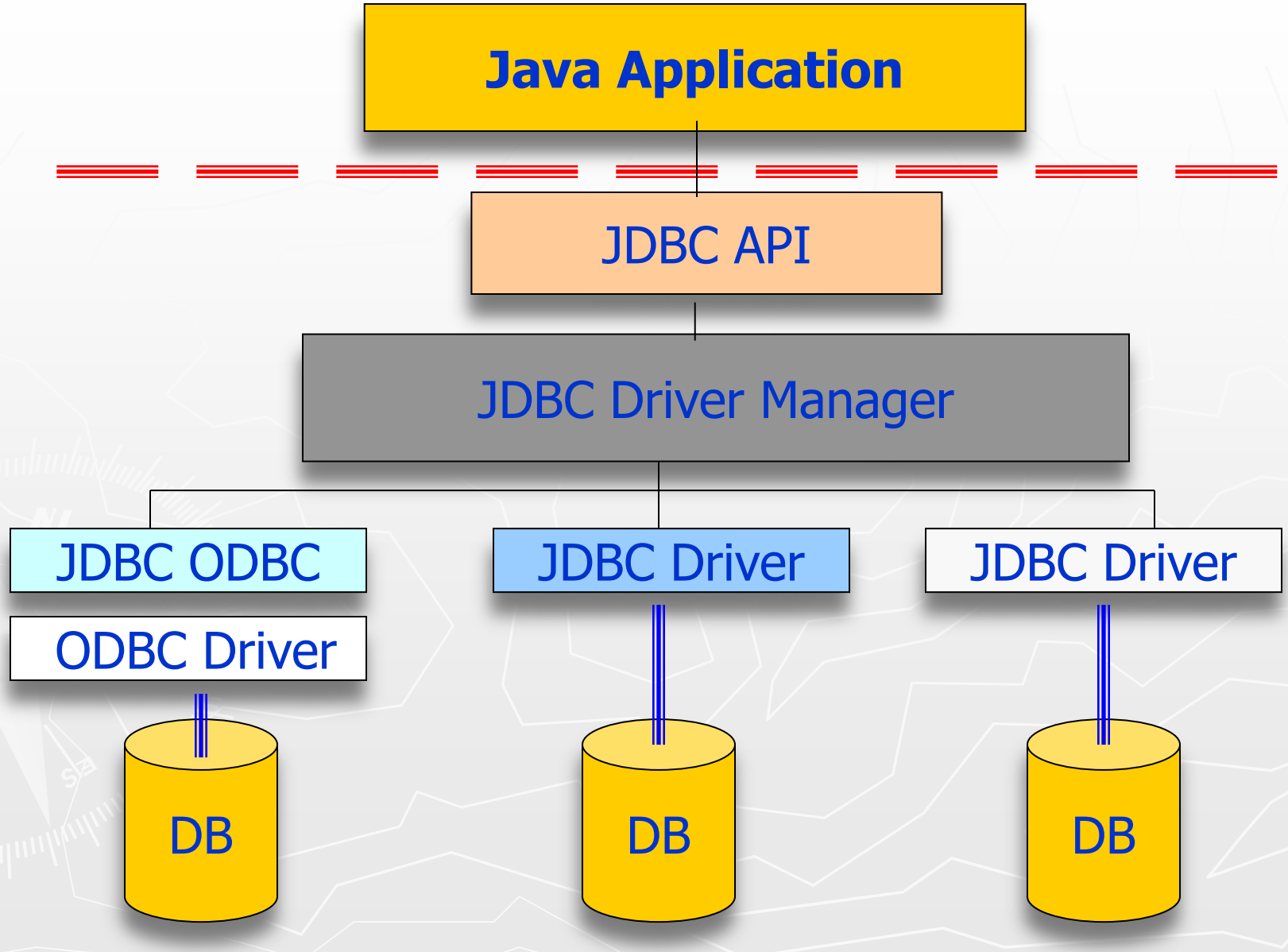
Database Connection History

- Pre-ODBC
  - Static SQL embedded inside a language such as Cobal, C..ect.
- ODBC
  - Call-level interface where sql statements are dynamically passed.
- JDBC
  - Platform independence.
  - Innovation within specification.

Why JDBC ?

- Why JDBC:
  - Different DBMS uses slightly different versions of SQL's.
  - Different DBMS uses different data types.
- The goals:
  - To provide Java programmers with a uniform, simple interface to a wide range of relational databases DB independence.
  - It can replace underlying database with minimal code impact.
  - It defines a common base on which higher level tools and interfaces can be built.

Java Application Using JDBC



The JDBC Framework

- JDBC API (in the java.sql package)
  - all the specific details of a database system are hidden from the developers
  - Java application developers write their code according to the JDBC interfaces
    (i.e. database vendor-neutral code)
- JDBC driver manager (a JDBC class)
  - java.sql.DriverManager (a Java class)
  - search a list of registered JDBC drivers
  - create a database connection

The JDBC Framework

- JDBC drivers (Java classes)
  - database vendor-supplied Java classes that implement the JDBC interfaces
  - e.g. mysql-connector-java-5.1.30-bin.jar
- JDBC-ODBC bridge (Java JDK)
  - Java access to any database via a database vendor's ODBC driver
  - useful for testing purpose

Some JDBC Interfaces

- Driver
- Connection
- Statement
- PreparedStatement
- CallableStatement
- ResultSet
- ResultSetMetaData
- DatabseMetaData

JDBC Drivers

- Java JDBC Driver types
  - type 1: JDBC/ODBC bridge (JDK)
    - for connection to any ODBC driver
    - In Sun's JDK package
  - type 2: JDBC native drivers (vendor-supplied)
    - the driver and the database must run on the same system
    - Java code that calls native C or C++ methods
  - type 3: JDBC net drivers (vendor- supplied)
    - Pure Java, for remote database access
    - use of a generic network API (i.e. database-independent)
  - type 4: JDBC thin drivers (vendor-supplied)
    - Pure Java, for remote database access
    - the network protocols are part of the database engine

JDBC Driver for Microsoft ODBC Data Source

- type 1 JDBC driver
  - sun.jdbc.odbc.JdbcOdbcDriver

JDBC Driver for MySQL

- type 4 JDBC driver
  - com.mysql.jdbc.Driver
  - database URL: "jdbc:mysql: <other stuff>"

- MySQL JDBC driver
  - Called: MySQL Connector/J
    - e.g. mysql-connector-java-5.1.30-bin.jar
  - Download:
    - http://dev.mysql.com/downloads/connector/j/5.1.html

JDBC Drivers for IBM DB2/400

- type 2: JDBC native driver (vendor-supplied)
  - **com.ibm.db2.jdbc.app.DB2Driver**
  - database URL: "jdbc:**db2**: <other stuff>"
- type 4: JDBC thin driver (vendor-supplied)
  - **com.ibm.as400.access.AS400JDBCDriver ****
  - database URL: "jdbc:**as400**: <other stuff>"

** Toolbox JDBC driver

JDBC Drivers for Oracle DB

- Oracle OCI Driver
  - = Type 2
- Oracle Thin Driver
  - = Type 4
- Oracle Internal Driver
  - Not any type, special to Oracle
  - Run inside the oracle's database JVM called kprb driver
- Oracle server-side Thin Driver
  - Not any type, special to Oracle
  - Used with java code that runs inside the oracle db JVM

Database Systems

- high-performance commercial databases
  - e.g. DB2, Oracle, MicroSoft SQL Server
- desktop databases
  - e.g. Access, Paradox
- open-source databases
  - e.g. MySQL, PostgresSQL
- lightweight Java database
  - e.g. Cloudscape, Pointbase, SQLite

A Simple JDBC Program

- `SimpleJDBC.java`

- Using MySQL Database on ZENIT
  - https://zenit.senecac.on.ca/info/ => MySQL Server
- Connecting to MySQL from command line:
  - To connect to MySQL account on Zenit from inside the Zenit cluster use the following command:

    $ mysql -u username -p -h db-mysql.zenit database
  - To connect to MySQL account on Zenit from outside the Zenit cluster use the following command:

    $ mysql -u username -p -h zenit.senecac.on.ca database

JDBC Programming

- Step 1: Register a JDBC driver.
- Step 2: Connect to the database.
- Step 3: Prepare SQL statement objects.
- Step 4: Execute SQL statement objects.
- Step 5: Retrieve information from the Result Sets.
- Step 6: Handle SQLException objects.
- Step 7: Close the statements, Result Sets and connection.

1. Register DB2 JDBC driver

- hard-coded registration

  DriverManager.registerDriver(new com.ibm.as400.access.AS400JDBCDriver( ));

- dynamic (i.e. run-time) registration

  Class.forName("COM.cloudscape.core.JDBCDriver");

  - use of a properties file

Register MySQL JDBC Driver

- Class.forName(“com.mysql.jdbc.Driver”);

2. Connect to the database

- database URL: “jdbc:<subprotocol>:<other stuff>”

```
Connection conn = DriverManager.getConnection (
      “jdbc:as400://zeus.senecac.on.ca/Bookstore”   );
```

  - subprotocol: as400 -
  - database server: zeus.senecac.on.ca
  - database name (collection name): Bookstore

Connect to the MySQL database

- database URL: "jdbc:<subprotocol>:<other stuff>"

```
Connection conn = DriverManager.getConnection (
    "jdbc:mysql://localhost/accounts"   );
```

  - subprotocol: mysql
  - database server: localhost
  - database name: accounts

3. Prepare SQL statement objects

- plain SQL Statements

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

- Prepared Statements
  - the query plan is built only once
  - prepareStatement( )

- Callable Statements
  - stored procedures precompiled in the database
  - prepareCall( )

About Prepared Statement

- The PreparedStatement class is derived from the more general class, Statement.
- If you want to execute a Statement object many times, it usually reduces execution time to use a PreparedStatement object instead.
- The main feature of a PreparedStatement object is that,
  - A Statement object send an SQL statement to the DBMS, where it is compiled each time.
  - A PreparedStatement object makes a SQL statement that precompiled. This means that when the PreparedStatement is executed, the DBMS can just run the PreparedStatement SQL statement without having to compile it first.

4. Execute SQL statement objects

- boolean execute (String sql)
  - any SQL statement (e.g. CREATE)
  - true is returned if a ResultSet is produced
  - multiple ResultSets may be produced
- ResultSet execute Query (String sql)
  - a ResultSet is produced by the SQL statement
  - a single ResultSet object is returned
- int execute Update (String sql)
  - executes an SQL INSERT, UPDATE or DELETE statement

5. Analyze/Navigate the Result Set

- next ( )
- the getters: SQL types => Java types
  - getString ( )
  - getBigDecimal ( )
  - getInt ( )
  - getDouble ( )
  - getDate ( )
  - getTime ( )

  ... [Tables 13.9, 13.10]
- JDBC Tutorial: Basics (http://java.sun.com/products/jdbc)

The Metadata of a Result Set

- the ResultSet MetaData interface
  - get MetaData ( )
  - get ColumnCount ( )
  - get ColumnLabel ( )
  - getr CoumnTypeName ( )
  - get Object ( )

6. Handle SQLException objects.

- JDBC API throws SQLException exceptions.

MySQL JDBC Example-2

- Java program that use JDBC to create database table – Account:
  - CreateDB.java
  - DBConnection.java
  - Account.sql
  - database.properties
    - Need to use the Properties class to read entries from database.properties.
- Using *if (stat.execute(sqlLine)) showResultSet(stat)* to execute each line of the Account.sql file
  - boolean execute(String sqlLine) method will
    - return false for non-query sql statement;
    - return true for sql query statement; then process the ResultSet.

MySQL JDBC Example-3

- Java program that use JDBC to query the created database table - Account:
  - QueryDB.java
  - DBConnection.java
  - database.properties

- int executeUpdate (String sql)
  - Used for SQL statement of insert, update and delete.
  - when returned integer is greater than or equal to 0, it indicates the number of rows which have been affected.

MySQL JDBC Example-4,5

- Java Program connect MySQL
  - **TestJDBC_mysql.java**

Java Program connect IBM System i

- TestJDBC_i.java

Reference

- The Java™ Tutorials: JDBC(TM) Database Access
  http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

- Java SE Technologies - Database
  http://www.oracle.com/technetwork/java/javase/jdbc/index.html

- JDBC Tutorial: Basics
  http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Thank You!