CMPUT 391

Database Management Systems

Lab: JDBC Review

Department of Computing Science
University of Alberta

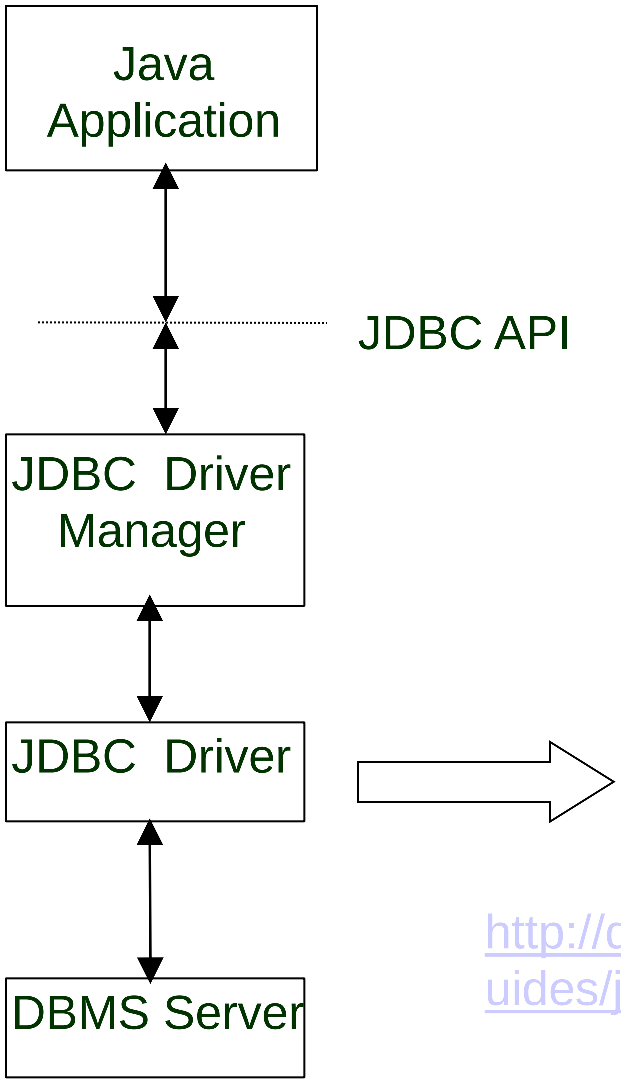
What Is JDBC?

- JDBC is a programming interface
- JDBC allows developers using java to gain access to a wide range of DBMSs
- JDBC allows users of different operating systems to access the same DBMS

What can you do with JDBC?

- Connect to different types of DBMSs
- Integrate SQL with Java to interact with data sources
- JSP, Servlet, Applet…
- Your Assignments and/or Projects
- ….

JDBC Architecture



Comes in four varieties

For more information:

http://download.oracle.com/javase/6/docs/technotes/guides/jdbc/getstart/intro.html 1.2.8 JDBC Driver Types

JDBC Interface

- Provides
  - – Library of function calls
  - – Standard way to connect
  - – Standard representation of data types
  - – Standard set of error codes
- Performs
  - – Connections with a database or other tabular data sources
  - – Sending of SQL statements
  - – Processing of results

Basic JDBC Classes

- DriverManager: manages connection between the data source and driver
- Connection: establishes connection to data source
- Statement: used to send DDL and DML statements to the data source
- ResultSet: used to access results of a query

Getting Started

- Environment Settings

```
> echo $CLASSPATH
```

If /oracle/jdbc/lib/classes12.zip is not set:

- For bash users, add to .bashrc:

```
> export CLASSPATH=$CLASSPATH\:.\:/oracle/jdbc/lib/classes12.zip
```

- For csh or tcsh users, add to .cshrc:

```
> setenv CLASSPATH $CLASSPATH\:.\:/oracle/jdbc/lib/classes12.zip
```

- Import java.sql package in your java code

```
import java.sql.*;
```

Getting Started (cont.)

- `oraenv` **and** `coraenv` **are Unix/Linux** command line utilities that set the required environment variables (ORACLE_SID, ORACLE_HOME and PATH) to allow a user to connect to a given database instance.
- If you are using bourne shell

  source /oracle/oraenv or

  if you are using the c shell,

  source /oracle/coraenv
- http://gwynne.cs.ualberta.ca/~oracle/

Getting Started (cont.)

- Establishing a connection
  - Loading the driver

    ```
    Class drvClass = Class.forName(driverName);
    ```

  - Making a connection

    ```
    Connection con = DriverManager.getConnection (url,
                                        userName, password);
    ```

  - driverName = oracle.jdbc.driver.OracleDriver
  - url = jdbc:oracle:thin:@gwynne.cs.ualberta.ca:1521:CRS
  - Calling Class.forName will create an instance of the driver and register it for you automatically. You don’t need to use DriverManager.registerDriver method in this context.
  - http://download.oracle.com/javase/6/docs/technotes/guides/jdbc/getstart/drivermanager.html

Getting Started (cont.)

- Sending SQL Statements
  - Creating JDBC Statements

    Statement stmt = m_con.createStatement();
  - Executing Statements

    stmt.executeQuery("select mname, category from movie");

    stmt.executeUpdate("insert into movie (movieid) values (101)");
- Manipulating the results: **ResultSet** class

ResultSet Class

- represents the results of a query to the database
- the result can be read sequentially from the **ResultSet object** using method **next()**

- Example:
  - ***rset.next()***, where ***rset*** is an object of class ***ResultSet***

Creating a ResultSet

➢ use method **createStatement** from class **Connection** to specify properties of the **ResultSet** object.

➢**Example:** creating a scrollable ResultSet

```
Statement stmt= conn.createStatement(
                    ResultSet.TYPE_SCROLL_SENSITIVE,
                    ResultSet.CONCUR_READ_ONLY);
ResultSet rset=stmt.executeQuery(“select …”);
```

ResultSet Properties Specification

- **types:**
  - ➢ TYPE_FORWARD_ONLY (default)
  - ➢ TYPE_SCROLL_INSENSITIVE (allows scroll and doesn’t reflect changes)
  - ➢ TYPE_SCROLL_SENSITIVE (reflects changes)
- **concurrency:**
  - ➢ CONCUR_READ_ONLY (default, no updates)
  - ➢ CONCUR_UPDATABLE (updates allowed)

ResultSet Methods for Cursors

- rset.next() – move forward one row;
- rset.previous() – move backward one row;
- rset.first() – move to the first row;
- rset.last() – move to the last row;
- rset.getRow() – obtain current position
- rset.absolute(int n) – move to the *n*th row
- rset.relative(int n) – move *n* rows from current
- …

Updating Data using ResultSet

- move **ResultSet *rset*** to the row to be changed
- use method **rset.updateXXX(column,value)** to change
  - e.g.: rset.updateInt(1,25) OR rset.updateInt(“age”,25)
- use method **rset.updateRow()** to make changes permanent

Note 1: statement must be **CONCUR_UPDATABLE**

Note 2: cannot be used if query is “select * from…”

Inserting Data using ResultSet

- move cursor to **special insert row** using method call **rset.moveToInsertRow()**
- set every column value using method call **rset.updateXXX(…)**
- insert new row in **ResultSet and table** using method call **rset.insertRow()**
- **Note:** statement must be **TYPE _SCROLL_SENSITIVE** and **CONCUR_UPDATABLE**

Deleting Data using ResultSet

- move cursor to the desired row
- delete row from **ResultSet and table** using method call **rset.deleteRow()**

Note: statement must be **TYPE_SCROLL_SENSITIVE** and **CONCUR_UPDATABLE**

Also check:

http://download.oracle.com/javase/6/docs/technotes/guides/jdbc/getstart/resultset.html

Using MetaData to Display the ResultSet

- **ResultSetMetaData** class provides information about types and properties in a ResultSet

- use method **getMetaData()** on a ResultSet object to get the result set's metadata information

- use methods from **ResultSetMetaData** class to get the available information
  - **getColumnCount ()** – returns the number of columns in the ResultSet
  - **getColumnLabel (int column)** – returns the column title for use to display
  - **getColumnType (int column)** – returns a column's SQL type

Using MetaData to Display the ResultSet (cont.)

```
…
ResultSetMetaData rsetMetaData = rset.getMetaData(); /* get metadata for ResultSet rset*/
int columnCount = rsetMetaData.getColumnCount(); /* get number of columns in result set*/
for ( int column = 1; column <= columnCount; column++) {…
        value = rsetMetaData.getColumnLabel(column); /* get column name */
… }
while (rset.next() ) {…
      for ( int index = 1; index <= columnCount; index++) {… /* get data one tuple at a time */
        o = rset.getObject(index); /* get the value as an Object */
        if (o != null )         value = o.toString(); /* convert it to String to display it */
        else                    value = "null";
        …}
…}
…
```

What's Next:

–Work through the JDBC examples posted on the Course Webpage

–Other reference materials can be found in the "Reference Materials" section of the course webpage

Questions?