Introduction to JDBC

Mark Allen Weiss
Copyright 1999

Outline of Topics

- Basic principles
- Making your database visible
- Java code

Basic Ideas

- Two layers
  - The JDBC API
  - JDBC Manager Driver API
- JDBC API communicates with manager using SQL statements.
- Manager communicates with various database drivers to translate the SQL to into database queries for the appropriate database.
- Database vendors should supply drivers; as a database user, you are only concerned with JDBC API.
SQL
- The standard database query language.
- JDBC requires support for SQL-92.
- If you know SQL, it is trivial to construct Java code to access a database.

Basic SQL Commands
- SELECT
- UPDATE
- DELETE
- INSERT INTO
- CREATE TABLE

SELECT Statements
- Basic Query Components
  - SELECT columns
  - FROM table
  - WHERE criteria
  - ORDER BY how to order
  - LIMIT number of rows
- FROM is required; others are optional
- columns can be * to list all columns, or comma-separated list of a subset of columns
Examples

- SELECT * FROM hockey
- SELECT name, goals, assists, points FROM hockey
  ORDER BY points DESC
- SELECT name, goals, assists, points FROM hockey
  ORDER BY points DESC LIMIT 40
- SELECT * FROM hockey WHERE goals > 20 AND
  assists > 20 AND points > 50 ORDER BY points

Database URLs

- A database URL looks like: `jdbc:subprotocol name:database url`
- Example:
  `jdbc:odbc:database.csv`
  `jdbc:odbc://data.ticketmaster.com:8888/db1;PWD=secret`
- odbc subprotocol is always available.

Connecting

- Need a driver manager to be loaded.
- Use `Class.forName` to load the driver manager class.
- For odbc, use
  `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`
- Once class is loaded, use static method
  `DriverManager.getConnection`. Provide a database URL, and optionally a name and password. This returns a Connection object.
Connection Interface

- Allows you to
  - create queries
  - get database meta-data
  - commit or rollback transactions
- Connection not made until later request.
- Important methods:
  - `Statement createStatement();`
  - `PreparedStatement prepareStatement(String sql);`
  - `void setAutoCommit(boolean autoCommit);`
  - `DatabaseMetaData getMetaData();`
  - `void rollback();`

Statements and ResultSets

- Statement is a query that can be sent to the database.
- Important methods:
  - `ResultSet executeQuery(String sql);`
  - `int executeUpdate(String sql);`
- The ResultSet contains an enumeration-type pattern; each item in the enumeration is a row in the result.
- Can get elements in the current row of the enumeration using `getXXX(int column)`. Note: columns begin at 1.

Typical Code

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

  String url = "jdbc:odbc:somedb";
  Connection conn = DriverManager.getConnection(url);
  Statement stmt = conn.createStatement();
  String sql = "SELECT Name FROM directory.csv";
  ResultSet r = stmt.executeQuery(sql);
  while (r.next()) {
    System.out.println(r.getString(1));
  }
} catch (Exception e) {
  e.printStackTrace();
}
```
Prepared Statements

- Useful for similar-looking repeated queries,
- `Connection.prepareStatement` gives you a prepared statement; provide a string with `?` to store the placeholders.
- Use `setXXX(whichPlaceHolder, value)` to set the placeholder in the prepared statement.
- Note that placeholder counting starts at 1.
- After placeholders filled, can call `executeQuery`.

Summary

- JDBC is an easy-to-use interface to databases.
- Hardest part is setting up the databases outside of Java.