CSC309 Tutorial: JDBC

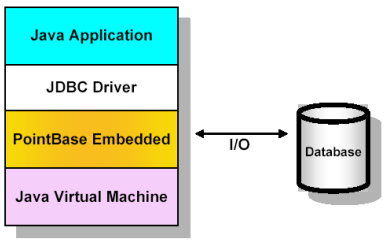
TA: Tianhan Wang
http://www.cs.toronto.edu/~tianhan/TA/CSC309S/

Outline

- JDBC and PointBase: Overview
- Using PointBase JDBC
- A Complete Example
- More Readings

JDBC and PointBase: an Overview

- The JDBC API (in java.sql package) is used by Java applications to access and manipulate the data stored in a database by invoking SQL commands.
- PointBase supports the JDBC API. For a list of supported and unsupported methods, refer to the PointBase documentation.
- Interaction of applications with the PointBase embedded:



Basics for Writing a JDBC Application

- Making a connection to PointBase
  - Loading the JDBC driver
  - Connecting to the PointBase database
- Creating and executing JDBC statements
  - Static
  - Dynamic
- Retrieving results
  - Scrollable
  - Non-scrollable
- Committing and closing objects
  - Committing or rolling back changes made
  - Closing result sets, JDBC statements and connections

Making a Connection to PointBase

- Assume we already have a database named "sample"
- Loading the PointBase JDBC driver

```
strDriver = "com.pointbase.jdbc.jdbcUniversalDriver";  //PointBase universal JDBC driver
Class.forName( strDriver ).newInstance( );  //Load the PointBase JDBC driver
```

- Connecting to the PointBase database

```
// Working with "sample" database
strURL = "jdbc:pointbase:embedded:sample";
strUserName = "public";    // default username
strPassword = "public";    // defualt password
_conn =DriverManager.getConnection( strURL, strUserName, strPassword );
                                //_conn is a Connection object
```


Creating/Executing Static JDBC Statements

- Define an SQL statement

```
String strCreateTable = "CREATE TABLE COFFEES "  +
                        "(COF_NAME VARCHAR(32)," +
                        " SUP_ID INTEGER,"       +
                        " PRICE FLOAT,"          +
                        " SALES INTEGER, "       +
                        " TOTAL INTEGER) ";
```

- Create a static SQL statement

```
_stmt = _conn.createStatement();  //_stme is an Statement object
```

- Execute a static SQL statement

```
_stmt.executeUpdate( strCreateTable );
```


Creating/Executing: More Examples

- Insert Data

```
_stmt.executeUpdate("INSERT INTO COFFEES " +
                    "VALUES ('French_Roast', 49, 8.99, 0, 0)");
```

- Delete Table

```
String strDelTable = "DROP TABLE COFFEES ";
_stmt.executeUpdate( strDelTable );  // deleting the table coffees
```

- Queries

```
String strQuery = "SELECT COF_NAME, PRICE FROM COFFEES";
ResultSet rs = _stmt.executeQuery( strQuery );
```


Retrieving Data from Non-scrollable RS

- A non-scrollable result set only allows you to retrieve the values stored in the result set in sequential order.
- When a result set is returned, the cursor is positioned before the first row of the result set. To access the first value of the result set you must advance the cursor to the first row using the resultSet.next() method. This method is used to move the cursor from row to row in the result set, and returns a Boolean TRUE value if there is data in the row to which the cursor is pointing.
- Retreiving data from tables

```
String strQuery = "SELECT COF_NAME, PRICE FROM COFFEES";
ResultSet rs = _stmt.executeQuery( strQuery );
while (rs.next()) {
                String s = rs.getString("COF_NAME");
                float n = rs.getFloat("PRICE");
                System.out.println(s + " " + n);
}
```

- Retrieve specific datatypes

getInt, getShort, getBoolean, getDouble, getFloat, getDate, getTime, getTimestamp

Closing and Committing Objects

- Rolling back or Committing the transaction

```
// Rollback any changes made to the database
// Use m_conn.commit() if you don't wish to rollback the transaction
m_conn.rollback();
```

- Closing the result set

```
rs.close()
```

- Closing the JDBC statement

```
_stmt.close()
```

- Closing the connection to the database

```
_conn.close()
```


More on PointBase JDBC

- Creating/Executing a dynamic JDBC statement

  A dynamic JDBC statement can improve performance of applications relative to static JDBC statements. Unlike a static JDBC statement, dynamic or prepared statements are only compiled once, regardless of the number of times that they are used.

- Creating/Using the result set of scrollable type

  By returning a scrollable type of result set, you have the capability to retrieve result set row values in any order. Conversely, using a non-scrollable result set, you can only retrieve result set row values as you scroll forward. With scrollable result sets, however, you can scroll either forward or backward. Additionally, you can also scroll by specifying a position in the result set.

- Using Transactions

  A transaction is a set of one or more statements that are executed together as a unit, so either all of the statements are executed, or none of the statements is executed.

- Stored Procedures

  A stored procedure is a group of SQL statements that form a logical unit and perform a particular task. Stored procedures are used to encapsulate a set of operations or queries to execute on a database server.

More on Creating JDBC Applications

- Import required classes: java.sql. *
- Using try/catch blocks to handle exceptions
- Retrieve exceptions: JDBC lets you see the warnings and exceptions generated by your DBMS and by the Java compiler. To see exceptions, you can have a catch block print them out.

```
try {
        // Code that could generate an exception goes here.
        // If an exception is generated, the catch block below  will print out information.
} catch(SQLException ex) {
        System.err.println("SQLException: " + ex.getMessage());
}
```


A Complete Example

```
import java.sql.*

public class DBTable  {

 protected Connection  _conn;
 protected Statement   _stmt;

 public DBTable(  ) {
 }

 public void init( ) throws Exception {
    String strDriver, strURL, strUserName, strPassword;
    strDriver = "com.pointbase.jdbc.jdbcUniversalDriver";
    Class.forName( strDriver ).newInstance( );

    // Working with "sample" database
    strURL = "jdbc:pointbase:embedded:sample";
    strUserName = "public";  // default username
    strPassword = "public";  // defualt password
    _conn  = DriverManager.getConnection( strURL, strUserName, strPassword);
    _stmt = _conn.createStatement();
 }
```

```
public void create( ) throws Exception {
    // Creating the table coffees
     String strCreateTable = "CREATE TABLE COFFEES "  +  "(COF_NAME VARCHAR(32)," +
                                 " SUP_ID   INTEGER,"    +   " PRICE    FLOAT,"      +
                                 " SALES    INTEGER, "   +   " TOTAL    INTEGER) ";
     _stmt.executeUpdate( strCreateTable );
 }

  public void insert( ) throws Exception  {
    // Inserting data into table Coffees
    _stmt.executeUpdate("INSERT INTO COFFEES " + "VALUES ('French_Roast', 49, 8.99, 0, 0)");
    _stmt.executeUpdate("INSERT INTO COFFEES " + "VALUES ('Espresso', 150, 9.99, 0, 0)");
   }

  public void query( ) throws Exception    {
    // Retreiving data from tables
    String strQuery = "SELECT COF_NAME, PRICE FROM COFFEES";
    ResultSet rs    = _stmt.executeQuery( strQuery );
    while (rs.next()) {
       String s = rs.getString("COF_NAME");
       float n = rs.getFloat("PRICE");
       System.out.println(s + "   " + n);
    }
    rs.close( );
  }
```



```
public void close( ) throws Exception {
    _stmt.close( );
    _conn.close( );
 }

 public void delete( ) throws Exception {
    // deleting the table coffees
    String strDelTable = "DROP TABLE COFFEES ";
    _stmt.executeUpdate( strDelTable );
 }

public static void main( String strarrArgs[] ) {
     try   {
              DBTable dbTable;
              dbTable = new DBTable( );
              dbTable.init( );
              dbTable.create( );
              dbTable.insert( );
              dbTable.query( );
              dbTable.delete( );
              dbTable.close( );
     }    catch( Throwable thr )  {
              thr.printStackTrace( );
     }
  }
}
```


More Readings

- Try the Examples
  http://www.cdf.toronto.edu/~t2aboels/samples/sample2/
- CDF Resources
  http://www.cdf.toronto.edu/~t2aboels/#JDBC
- JDBC Documentation and Tutorials
  http://java.sun.com/products/jdbc/
- PointBase Developer's Guide
  http://www.pointbase.com/support/docs/pbdeveloper.pdf