APPENDIX

Building a Stock Database

This example demonstrates how to work with the *Record Management System (RMS)* and the *Generic Connection Framework* and *HttpConnection* to build a real MIDlet application. The MIDlet for this example does the following:

- Creates a record store (database)
- Adds new records (stocks) to the database)
- Views the stocks in the database.

To add a stock to the database, the user enters the stock symbol (such as, SUNW, IBM, MOT, AOL, etc.). The MIDlet retrieves the corresponding stock quote from the Yahoo Quote Server (http://quote.yahoo.com), constructs a record, and adds the record to the database.

To view the stocks in the record store, the MIDlet iterates through the records and prints them on the display in a nice format.

Implementation

The implementation consists of the following three classes: *Stock.java*, *StockDB.java*, and *QuotesMIDlet.java*.

The Stock.java Class

This class parses a string obtained from the yahoo Quote Server or the record store into fields (such as name of stock or price). The string obtained from the Yahoo Quote Server has the following format:

Name Time Price Change Low High OPEN PREV
"SUNW" "2:1PM - 79.75", +3.6875, "64.1875 - 129.3125", 78, 76.0625

In this MIDlet, the fields retrieved are the name of the stock, the time, and the price.

Stock.java

```
public class Stock {
 private static String name, time, price;
   // Given a quote from the server, retrieve the name, price, and
   // date of the stock
 public static void parse(String data) {
   int index = data.indexOf('"');
   name = data.substring(++index, (index = data.indexOf('"', index)));
   index +=3;
   time = data.substring(index, (index = data.indexOf('-', index))-1);
   index +=5;
   price = data.substring(index, (index = data.indexOf('<', index)));
 }

 // Get the name of the stock from the record store
 public static String getName(String record) {
   parse(record);
```



```
    return(name);
  }

  // Get the price of the stock from the record store
  public static String getPrice(String record) {
    parse(record);
    return(price);
  }
}
```

The StockDB.java Class

This class provides methods that do the following:

- Open a new record store
- Adds a new record to the record store
- Closes the record store
- Enumerates through the records

Once you understand how to open a record store, add a new record, and close the record store, this code is easy to follow.

StockDB.java

```
import javax.microedition.rms.*;
import java.util.Enumeration;
import java.util.Vector;
import java.io.*;

public class StockDB {

   RecordStore recordStore = null;

   public StockDB() {}

   // Open a record store with the given name
   public StockDB(String fileName) {
      try {
        recordStore = RecordStore.openRecordStore(fileName, true);
      } catch(RecordStoreException rse) {
        rse.printStackTrace();
      }
   }

   // Close the record store
   public void close() throws RecordStoreNotOpenException,
                              RecordStoreException {
        if (recordStore.getNumRecords() == 0) {
            String fileName = recordStore.getName();
            recordStore.closeRecordStore();
            recordStore.deleteRecordStore(fileName);
        } else {
            recordStore.closeRecordStore();
        }
    }

   // Add a new record (stock) to the record store
   public synchronized void addNewStock(String record) {
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
```



```
      DataOutputStream outputStream = new DataOutputStream(baos);
      try {
          outputStream.writeUTF(record);
      }
      catch (IOException ioe) {
          System.out.println(ioe);
          ioe.printStackTrace();
      }
      byte[] b = baos.toByteArray();
      try {
          recordStore.addRecord(b, 0, b.length);
      }
      catch (RecordStoreException rse) {
          System.out.println(rse);
          rse.printStackTrace();
      }
    }

    // Enumerate through the records.
    public synchronized RecordEnumeration enumerate() throws
RecordStoreNotOpenException {
       return recordStore.enumerateRecords(null, null, false);
    }
}
```

The QuotesMIDlet.java Class

The QuotesMIDlet class is the actual MIDlet that does the following:

- Creates commands (List Stocks, Add New Stock, Back, Save, Exit)
- Handles command events
- Connects to the Yahoo Quote Server and retrieves quotes
- Invokes methods from Stock and StockDB to parse quotes and add new stocks to the record store

QuotesMIDlet.java

```
import javax.microedition.rms.*;
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
import javax.microedition.io.*;
import java.io.*;
import java.util.Vector;

public class QuotesMIDlet extends MIDlet implements
CommandListener {
    Display display = null;
    List menu = null; // main menu
    List choose = null;
    TextBox input = null;
    Ticker ticker = new Ticker("Database Application");
  String quoteServer =
              "http://quote.yahoo.com/d/quotes.csv?s=";
    String quoteFormat = "&f=slc1wop"; // format supported

    static final Command backCommand = new Command("Back",
                                          Command.BACK, 0);
    static final Command mainMenuCommand = new Command("Main",
```



```
                                          Command.SCREEN, 1);
    static final Command saveCommand = new Command("Save",
                                          Command.OK, 2);
    static final Command exitCommand = new Command("Exit",
                                          Command.STOP, 3);
    String currentMenu = null;

    // Stock data
    String name, date, price;

    // record store
    StockDB db = null;

    public QuotesMIDlet() { // constructor
    }

    // start the MIDlet
    public void startApp() throws MIDletStateChangeException {
      display = Display.getDisplay(this);
      // open a db stock file
      try {
        db = new StockDB("mystocks");
      } catch(Exception e) {}
      menu = new List("Stocks Database", Choice.IMPLICIT);
      menu.append("List Stocks", null);
      menu.append("Add A New Stock", null);
      menu.addCommand(exitCommand);
      menu.setCommandListener(this);
      menu.setTicker(ticker);

      mainMenu();
    }

    public void pauseApp() {
      display = null;
      choose = null;
      menu = null;
      ticker = null;

      try {
        db.close();
        db = null;
      } catch(Exception e) {}
    }

    public void destroyApp(boolean unconditional) {
      try {
        db.close();
      } catch(Exception e) {}
      notifyDestroyed();
    }

    void mainMenu() {
      display.setCurrent(menu);
      currentMenu = "Main";
    }

    // Constuctor a running ticker with stock names and prices
```



```
  public String tickerString() {
     StringBuffer ticks = null;
     try {
        RecordEnumeration enum = db.enumerate();
        ticks = new StringBuffer();
        while(enum.hasNextElement()) {
          String stock1 = new String(enum.nextRecord());
          ticks.append(Stock.getName(stock1));
          ticks.append(" @ ");
          ticks.append(Stock.getPrice(stock1));
          ticks.append("    ");
        }
     } catch(Exception ex) {}
     return (ticks.toString());
  }

  // Add a new stock to the record store by calling
  //StockDB.addNewStock()
  public void addStock() {
    input = new TextBox("Enter a Stock Name:", "", 5,
                        TextField.ANY);
    input.setTicker(ticker);
    input.addCommand(saveCommand);
    input.addCommand(backCommand);
    input.setCommandListener(this);
    input.setString("");
    display.setCurrent(input);
    currentMenu = "Add";
  }

  // Connect to quote.yahoo.com and retrieve the data for
  // a given stock symbol.
  public String getQuote(String input) throws IOException,
                                NumberFormatException {
    String url = quoteServer + input + quoteFormat;
    StreamConnection c =
 (StreamConnection)Connector.open(url, Connector.READ_WRITE);
    InputStream is = c.openInputStream();
    StringBuffer sb = new StringBuffer();
    int ch;
    while((ch = is.read()) != -1) {
      sb.append((char)ch);
    }
    return(sb.toString());
  }

  // List the stocks in the record store
  public void listStocks() {
      choose = new List("Choose Stocks", Choice.MULTIPLE);
      choose.setTicker(new Ticker(tickerString()));
      choose.addCommand(backCommand);
      choose.setCommandListener(this);
    try {
       RecordEnumeration re = db.enumerate();
       while(re.hasNextElement()) {
         String theStock = new String(re.nextRecord());
         choose.append(Stock.getName(theStock)+" @ " +
                       Stock.getPrice(theStock), null);
```



```
            }
         } catch(Exception ex) {}
         display.setCurrent(choose);
         currentMenu = "List";
      }

      // Handle command events
      public void commandAction(Command c, Displayable d) {
         String label = c.getLabel();
         if (label.equals("Exit")) {
            destroyApp(true);
         } else if (label.equals("Save")) {
             if(currentMenu.equals("Add")) {
                 // add it to database
               try {
                 String userInput = input.getString();
                 String pr = getQuote(userInput);
                 db.addNewStock(pr);
                 ticker.setString(tickerString());

               } catch(IOException e) {
               } catch(NumberFormatException se) {
               }
               mainMenu();
             }

         } else if (label.equals("Back")) {
             if(currentMenu.equals("List")) {
               // go back to menu
               mainMenu();
             } else if(currentMenu.equals("Add")) {
               // go back to menu
               mainMenu();
             }

         } else {
            List down = (List)display.getCurrent();
            switch(down.getSelectedIndex()) {
              case 0: listStocks();break;
              case 1: addStock();break;
            }

         }
      }
   }
```

