JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 10, November-December 2007

Data Mining Historic Stock Quotes in Java

By **Douglas Lyon**

Abstract

This paper describes how to extract historic stock quote data and display it. We show how to chart stock prices, using the web as a data source.

The methodology for converting the web data source into internal data structures is based on using HTML as input. This type of screen scraping has become a popular means of recording data into programs. Simple string manipulation techniques are shown to be good enough for these types of well-formed data streams.

1 THE PROBLEM

Historic stock price data is generally available on the web (using a browser to format the HTML data). Given an HTML data source, we would like to find a way to create an underlying data structure that is type-safe and well formulated.

We are motivated to study these problems for a variety of reasons. Firstly, for the purpose of conducting empirical studies, entering the data into the computer by hand is both error-prone and tedious. We seek a means to get this data, using free data feeds, so that we can build dynamically updated displays and perform data mining functions. Secondly, we find that easy to parse data enables us to teach our students the basic concepts of data mining. This example is used in a first course in network programming.

2 FINDING THE DATA

Finding the data, on-line, and free, is a necessary first step toward this type of data mining. Quotes have been available, for years, from Yahoo. We can obtain the quotes, using comma-separated values (CSV) by constructing a URL and entering it into a browser. For example:

```
http://table.finance.yahoo.com/table.csv?s=IBM&a=00&b=2&c=1800
&d=04&e=8&f=2005&g=v&ignore=.csv
```

Cite this column as follows: Douglas A. Lyon “Data Mining Historic Stock Quotes in Java”, in *Journal of Object Technology*, vol. 6. no. 10, November-December 2007, pp. 17-23 http://www.jot.fm/issues/issue_2007_11/column2/

This creates an output on the screen that looks like:

```
Date,Dividends
2005-05-06,0.200000
2005-02-08,0.180000
2004-11-08,0.180000
2004-08-06,0.180000
2004-05-06,0.180000
…
```

Thus, we are able to obtain a listing of all the dividends that IBM paid, between two given dates. As another example, consider the week leading up to the IPhone release from Apple:

http://ichart.finance.yahoo.com/table.csv?s=aapl&a=5&b=22&c=2007&d=5&e=29&f=2007&g=d&ignore=.csv

The quotes are returned in the form:

```
Date,Open,High,Low,Close,Volume,Adj Close
2007-06-29,121.97,124.00,121.09,122.04,40513400,122.04
2007-06-28,122.36,122.49,120.00,120.56,29933700,120.56
2007-06-27,120.61,122.04,119.26,121.89,34810600,121.89
2007-06-26,123.98,124.00,118.72,119.65,47913500,119.65
2007-06-25,124.19,125.09,121.06,122.34,34478700,122.34
2007-06-22,123.85,124.45,122.38,123.00,22567000,123.00
```

The URL is decoded as:

s - ticker symbol
a - start month
b - start day
c - start year
d - end month
e - end day
f - end year
g - resolution (e.g. 'd' is daily, 'w' is weekly, 'm' is monthly)

To synthesize the URL needed to get the data, we use:

```
/**
     * Construct the URL necessary to retrieve all the quotes
for the given symbol between
     * the given dates from Yahoo.
     *
     * @param symbol the symbol to retrieve
     * @param start  the start date to retrieve
     * @param end    the end date to retrieve
     * @return URL string
     */
    public static String constructURL(String symbol, Calendar
start, Calendar end) {
        return "http://ichart.finance.yahoo.com/table.csv" +
"?s=" +
                symbol + "&a=" +
                Integer.toString(start.get(Calendar.MONTH) ) +
"&b=" +
```

```
                start.get(Calendar.DAY_OF_MONTH) + "&c=" +
                Integer.toString(start.get(Calendar.YEAR)) +
"&d=" +
                Integer.toString(end.get(Calendar.MONTH) ) +
"&e=" +

Integer.toString(end.get(Calendar.DAY_OF_MONTH)) + "&f=" +
                Integer.toString(end.get(Calendar.YEAR)) +
"&g=d&ignore=.csv";
    }
```

In order to fetch the data, given the URL, we write a simple helper utility that returns CSV data:

```
public static String[] getEodCSVData(String symbol,
                                              GregorianCalendar
gc2,
                                              GregorianCalendar
gc) {
        String urlString = constructURL(symbol, gc2, gc);
        System.out.println(urlString);
        return UrlUtils.getUrlString(urlString);
    }
```

The goal of such a program is to convert the URL into text, with one string per line, as retrieved from the web page. This is the core of the data retrieval phase.

3 ANALYSIS

In order to process the CSV data we need to decide how we are going to store and parse the data. To store the end-of-day quote we create an *EODQuote* class, with appropriate getters and setters:

```
public class EODQuote {
    // Date,Open,High,Low,Close,Volume,Adj. Close*
    //20-Sep-05,52.99,53.81,52.92,53.19,29279600,53.19
    private Date date;
    private double open;
    private double high, low, close, volume, adjustedClose;

    public EODQuote(String s) throws ParseException {
        StringTokenizer st = new StringTokenizer(s, ",");
        setDate(DateUtils.getEuropeanDate(st.nextToken()));
        setOpen(Double.parseDouble(st.nextToken()));
        setHigh(Double.parseDouble(st.nextToken()));
        setLow(Double.parseDouble(st.nextToken()));
        setClose(Double.parseDouble(st.nextToken()));
        setVolume(Double.parseDouble(st.nextToken()));
        setAdjustedClose(Double.parseDouble(st.nextToken()));
    }
```

To store multiple quotes, we create a container class that has high-level quote processing methods:

```
public class EODQuotes {
     private Vector v = new Vector();


    public EODQuotes(EODQuote[] e) {
        for (int i = 0; i < e.length; i++)
            add(e[i]);
    }

    public void add(EODQuote vr) {
        v.addElement(vr);
    }
    public EODQuote[] getRecords() {
        EODQuote vra [] = new EODQuote[v.size()];
        v.copyInto(vra);
        return vra;
    }
```

Now if we want to get the close prices, we can use:

```
public double[] getClosePrices() {
        EODQuote quotes[] = getRecords();
        double d[] = new double[quotes.length];
        for (int i = 0; i < d.length; i++)
            d[i] = quotes[i].getClose();
        return d;
    }
```

Note that the symbol does not appear in either the *EODQuote* or in the *EODQuotes*. Our assumption is that the quotes in the quote container are homogeneous in the sense that they all come from the same stock. If this is not the case, the data processing that results will be hard to interpret (or worse).

At this point, we build a mechanism for processing the data. For example, here is a simple moving average, with a period, measured in samples (in this case, trading days):

```
public double [] getSmaNew(int period) {
        this.sortByDateAscending();
        double[] in = getClosePrices();
        double sma[] = new double [in.length - period];
        for (int i = 0; i < sma.length; i++) {
            sma[i] = Stats.sma(in, i, period);
        }
        return sma;
    }
```

Where the average for a single window is given by:

```
/**
     *
     * @param sampleData
     * @param startPoint - offset into the sample data
     * @param period  - interval of the average
     * @return
     */
    public static double sma(double sampleData[], int
startPoint, int period) {
        double sum = 0;
```

```
            for (int i = startPoint; i < startPoint+period; i++)
                sum += sampleData[i];
            return sum / period;
        }
```

4 DISPLAY

In order to display our data, we use a graphing framework (whose description is beyond the scope of this paper;

```
    public static void main(String[] args) throws ParseException {
            System.out.println("Historical Quote Grapher Version
    1.0");
            String[] data = YahooEODQuotes.getEodCSVData(
                    "aapl",
                    DateUtils.getCalendar("06/22/2007"),
                    DateUtils.getCalendar("06/29/2007"));
            PrintUtils.print(data);
            EODQuotes appleData = new EODQuotes(data);
            double[] prices = appleData.getClosePrices();
            PrintUtils.print(prices);
            Graph.graph(prices, "June 22-29,2007", "aapl");
        }
```

We are interested in a new “killer application” for development, called the *JAddressBook* program. This program is able to chart historic stock quotes (and manage an address book, dial the phone, print labels, do data-mining, etc.). The program can be run (as a web start application) from:

http://show.docjava.com:8086/book/cgij/code/jnlp/addbk.JAddressBook.Main.jnlp

And provides an interactive GUI for charting stock data.

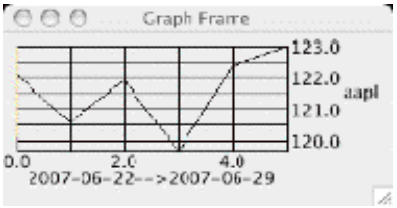

Figure 4-1. The Stock Chart

Figure 4-1 shows an image of the chart. The mining of CSV data is not new, our approach to graphing stock data in Java may be [Lyon 04D].

5 IMPLEMENTING A *STOCK CHART*

The stock chart is created using a graphing framework that makes use of the façade design pattern to simplify the interface:

```
/**
 * EODQuotes.graphGui -
 * prompts the user for symbol, start date and end date.
 */
    public static void graphGui() throws ParseException {
        System.out.println("Historical Quote Grapher Version
1.0");
        String symbol = In.getString("enter symbol");
        GregorianCalendar start = DateUtils.getCalendar();
        GregorianCalendar end = DateUtils.getCalendar();
        EODQuotes appleData1 = new
EODQuotes(YahooEODQuotes.getEodCSVData(
                symbol,
                start,
                end));
        EODQuotes appleData = appleData1;
        double[] prices = appleData.getClosePrices();
        PrintUtils.print(prices);
        Graph.graph(prices,
                DateUtils.getISO8601_String(start) +
                        "-->" +
DateUtils.getISO8601_String(end), symbol);
    }
```

This program prints the following on the console:

```
Historical Quote Grapher Version 1.0
http://ichart.finance.yahoo.com/table.csv?s=aapl&a=5&b=22&c=20
07&d=5&e=29&f=2007&g=d&ignore=.csv
122.04 120.56 121.89 119.65 122.34 123.0
```

6 CONCLUSION

The number of options involved in graphing data is responsible for creating large and complex frameworks. Our example provides a means of formatting the start and end dates in accordance with the ISO8601 standard, and we have made use of this standard for the display of the data. The question of how to best get symbol and date data from the user remains open. For our example, we simply prompt the user for a string and leave the detail of creating complex looking calendar dialogs for another time.

We showed how an ad-hoc parsing technique can be used to obtain CSV data from the web. The web is a huge and growing source of data. We rely upon others to keep the URL protocol and data consistent.

So far, Yahoo is a reliable and free source of historic stock data. Decoding the URL is often a challenge, and, in particular, the historic stock data has many retrieval options.

There are many sources of financial data in CSV format on the web. The question of what new uses we will find for this data remains open.

7 LITERATURE CITED

[Lyon 04D] *Java for Programmers*, by Douglas A. Lyon, Prentice Hall, Englewood Cliffs, NJ, 2004.

About the author

Douglas A. Lyon (M'89-SM'00) received the Ph.D., M.S. and B.S. degrees in computer and systems engineering from Rensselaer Polytechnic Institute (1991, 1985 and 1983). Dr. Lyon has worked at AT&T Bell Laboratories at Murray Hill, NJ and the Jet Propulsion Laboratory at the California Institute of Technology, Pasadena, CA. He is currently the Chairman of the Computer Engineering Department at Fairfield University, in Fairfield CT, a senior member of the IEEE and President of DocJava, Inc., a consulting firm in Connecticut. Dr. Lyon has authored or co-authored three books (Java, Digital Signal Processing, Image Processing in Java and Java for Programmers). He has authored over 30 journal publications. Email: lyon@docjava.com. Web: http://www.DocJava.com.