Visual Mining, Inc.

Using NetCharts™ with Java Server Pages and ODBC

A Programmers Guide to Scripting with JSP and NetCharts

Table of Contents

1. Scope

This document provides web page designers with detailed information on the capabilities of NetCharts when used with Java Server Pages. A companion document, *The Visual Mining CDL Reference Guide*, provides additional useful information on designing chart templates to be used with NetCharts.

Note to our customers:

Thank you for evaluating and/or purchasing NetCharts. We sincerely believe that the charts produced by NetCharts are among the most robust online charts available.

Please direct any questions or comments on this product to support@visualmining.com.

—The Visual Mining Team

2. Introduction

The purpose of this document is to provide web designers familiar with Sun's Java Server Pages technology an example of using NetCharts with JSP. An example will be given that: walks the user through installing an ODBC data source, runs the sample JSP script that is provided, and explains the script and how NetCharts and JSP interact with each other.

3. NetCharts JSP Example

Using NetCharts with JSP

This example uses JSP scripting commands to extract data from an ODBC data source (`regionalsales`), populate variables with the data, and build an HTML page that contains a NetCharts applet that uses this data. This paper provides a brief overview of how to use JSP to interact with ODBC, how to construct an HTML page containing a NetCharts applet, and how to populate that applet using NetCharts `<param>` tags and JSP variables.

Setting up an ODBC Data Source

The installation of NetCharts 4.0 should have added a small Access database to the host machine's ODBC Data Sources manager (this is assuming you are running on a Windows-based machine). It should be registered as a System DSN called `regionalsales`.

Alternatively, This ODBC data source can be configured manually using the following steps.

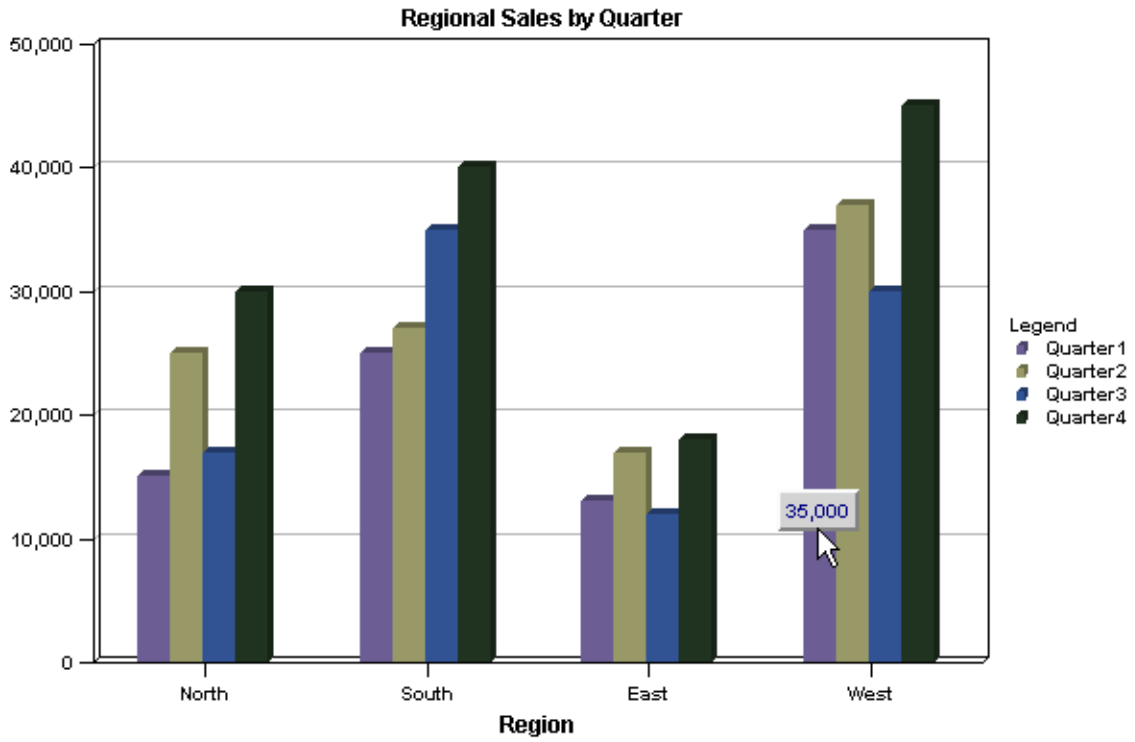
1) Launch the Control Panel.
2) Select Data Sources (ODBC).
3) Select the System DSN tab.
4) Add a "Microsoft Access Driver (`*.mdb`)" data source:
   a) Name the data source `regionalsales.`
   b) Select the regionalsales.mdb file. Its default location is `c:\program files\visual mining\netcharts4.0\netcharts\examples.`

Running the Example

The example can be configured and run using the following steps:

1. Make sure that regionalsales.mdb is available as a System DSN (see above).
2. Copy the `getSalesDataChart.jsp` file to any subdirectory under the JSP-enabled Web Application Server. For instance, when using JRun 3.0, it could be in the `c:\program files\allaire\jrun\servers\default\default-app` directory or any subdirectory the \default-app directory (assuming you are running this Java Server Page under the default application directory).
3. Make a copy of the NetCharts 4.0 `netcharts.jar` file under this same directory. The jar file can be found in the \classes directory under the NetCharts 4.0 installation directory. The default installation directory for NetCharts 4.0 is c:\program files\visual mining\netcharts 4.0\netcharts\.
4. Start a web browser and load the JSP page. For example, if the JSP page was installed as in steps 2 and 3 above, the URL would look like `http://localhost:8100/getSalesDataChart.jsp.` This points to the root directory of the JRun installation, which is listening on port 8100 of the host machine.

When you load the page, you should see something like:



Looking at the Code

The basic sequence of events in using JSP to populate a NetCharts applet is as follows:

- Define the variables
- Connect to the database and pass in the SQL statement
- Open the database and extract the data
- Populate the variables with the data from the database
- Close the connection to the database
- Instantiate the applet and pass the variables in through the NFParamScript parameter.

The following code fragments demonstrate how this is accomplished.

The first section of code defines several variables. These variables are used to connect to the database, pass in the SQL statement, and hold the values in string format to be passed to the NetCharts applet:

```
// variables for connecting to the ODBC data source
String driverName = "sun.jdbc.odbc.JdbcOdbcDriver";
String serverURLBase = "jdbc:odbc:";
String dbName = "regionalsales";
String sql = "Select region, quarter1, quarter2, quarter3, quarter4 from
Sales";

// variables for holding the data from the data source
String ds1 = "25,50,75,100";
String ds2 = "100,200,300,400";
String ds3 = "100,125,150,200";
String ds4 = "100,75,50,25";
String labels = "\"North\", \"South\", \"East\", \"West\"";

Connection connection = null;
```

```
Statement statement;
boolean isdata;
ResultSet rs;
int updateCount = 0;
```

The next section of code loads the JDBC driver, opens a connection to the database, and prepares and executes the SQL statement.

```
try {
      Class.forName(driverName);
} catch (Exception e){
}

try {
      // Open connection to db
      connection = DriverManager.getConnection(serverURLBase+dbName);

      // create statement to handle SQL
      statement = connection.createStatement();
      isdata = statement.execute(sql);
```

Next, the code loops through the ResultSets extracted from the database. The resulting data is placed into strings. These strings contain the data in a comma-separated format, which NetCharts can understand. Ultimately, the data will look something like “5,10,15,20”. After each row of data is extracted, the ResultSet is closed. After all of the data is extracted, the Statement and Connection are closed.

```
      int loopCount=0;
      // This loop handles multiple sql statements or stored procedures
      // that return multiple result sets.
      do {
            loopCount++;
            if (loopCount > 50){
                  // This prevents errors, like with the Symantec
                  // driver, always returning an update count of 0
                  break;
            }

            rs = statement.getResultSet();
            ResultSetMetaData rsmd = rs.getMetaData();
            int numColumns = rsmd.getColumnCount();

            boolean isdatainrow;
            ds1 = ds2 = ds3 = ds4 = labels = "";
            for (int rowNum=0; rs.next(); rowNum++){
                  Vector row = new Vector();

                  if (rowNum > 0)
                  {
                        ds1 += ",";
                        ds2 += ",";
                        ds3 += ",";
                        ds4 += ",";
                        labels += ",";

                  }

                  Object o = null;
                  for (int i=1; i <= numColumns; i++){
                        // we would return the actual object, but
                        // the JDBC-ODBC bridge crashes. :-(
```

```
                        o = rs.getString(i);
                        row.addElement(o);
                    }

                    labels += (String) row.elementAt(0);
                    ds1 += (String) row.elementAt(1);
                    ds2 += (String) row.elementAt(2);
                    ds3 += (String) row.elementAt(3);
                    ds4 += (String) row.elementAt(4);
              }
              rs.close();
              System.gc();
       } while (statement.getMoreResults() == true);

       // close the statement to clean up cursors.
       statement.close();
       if (!connection.isClosed())
              connection.close();
```

The applet tag can now be constructed. NetCharts uses an applet parameter called NFParamScript to pass in chart definition strings. A simple *name=value* format is used to construct complete chart definitions that can be processed by the NetCharts applet.

```
<applet name=Quarterly Sales
        code=NFBarchartApp.class
        codebase=/classes
        width=600 height=400>
<param name=NFParamScript value='
#Populate the chart with all of the static template information;
ChartName             = "Basic Grouped Barchart";
DebugSet                = ALL;
ChartWidth            = 600;
ChartHeight           = 400;
Background            = (white,NONE,3,null,TILE,black);
...
```

The relevant parameters for this chart are `DataSet1`, `DataSet2`, `DataSet3`, `DataSet4`, and `BarLabels`:

```
#Now populate the chart with the dynamic data extracted from the database via
JSP;
DataSet1                = <%=ds1%>;
DataSet2                = <%=ds2%>;
DataSet3                = <%=ds3%>;
DataSet4                = <%=ds4%>;
BarLabels               = <%=labels%>;
```

When this JSP page runs, it will convert the variables into the strings created earlier. These strings will then be passed in to the applet, and the chart will be created.

For a complete code listing, go to `/examples/getSalesDataChart.jsp`.

Visual Mining, Inc.
15825 Shady Grove Road
Suite 20
Rockville, MD 20850

Phone:
800-308-0731 in US
+1-301-795-2200 outside US

E-mail:
sales@visualmining.com
support@visualmining.com

Web:
http://www.visualmining.com