Visual Mining, a division of Tervela, Inc.

Programming with NetCharts® Pro 7.2

A Programmers Guide to Building Chart-Enabled Applications with NetCharts® Pro 7.2
Version 7.2
Summer 2015

Table of Contents

1.0 Scope

This document provides programmers with detailed information on the features, benefits, and architecture of NetCharts® Pro. NetCharts Pro is a is a comprehensive Java programmer-friendly chart library, allowing application developers to create award-winning data visualizations in virtually any format for interactive charting within applications, web-browsers, and internet-enabled devices without the need for a third party plug-in.

A companion document, *The Visual Mining Chart Definition Language Reference Guide*, provides additional useful information on designing chart templates to be used with NetCharts Pro. NetCharts Designer, can also be used to develop chart templates to be used with NetCharts Pro. More information about NetCharts Designer can be found at http://www.visualmining.com/nc-designer/.

Additional information on Java is available from https://www.oracle.com/java/index.html.

Note to our customers:

Thank you for evaluating and/or purchasing NetCharts Pro Version 7.2. We sincerely believe that the charts and chart images produced by NetCharts Pro 7.2 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*

2.0 Introduction

NetCharts Pro is one of a suite of data visualization solutions offered by Visual Mining. All of the products are written entirely in Java and use some form of Visual Mining's *Chart Definition Language* (CDL) as a basis for defining and generating charts. Because of this common base rendering engine, all NetCharts solutions are capable of generating the same charts. The products differ primarily in the way in which each is integrated into the software infrastructure of a user's application.

The Visual Mining Suite of Chart and Dashboard Generation Solutions

NetCharts Pro

NetCharts Pro (http://www.visualmining.com/nc-pro/) is a Java programmer-friendly chart generation solution. The NetCharts Pro API allows it to be used in Integrated Development Environments (IDE) such as Eclipse and IBM's WebSphere Studio. NetCharts Pro can create images of charts in standard web formats such as PNG, SVG and JPG, making it ideally suited for server-side use to chart enable applications using EJBs, servlets or JSP pages.

NetCharts Designer

NetCharts Designer (http://www.visualmining.com/nc-designer/) provides a comprehensive desktop Integrated Development Environments (IDE) for creating and managing charts, graphs, tables, interactive dashboards, and scorecards that can then be used in web-based applications. NetCharts Designer streamlines data analytics and development with one simple interactive dashboard solution. NetCharts Designer can be used to create chart templates for use by NetCharts Pro and complete dashboard applications for deployment within NetCharts Server.

NetCharts Server

NetCharts Server (http://www.visualmining.com/nc-server/) is a platform that can present complete dashboards or charts that developers define and publish using NetCharts Designer, the solution IDE. Together they are a traditional dashboard development solution allowing users complete control over the content, styling and interactivity included in the dashboards to implement very complex and rich dashboards. The NetCharts Server platform can also be used in conjunction with an entire range of web infrastructures from the simplest CGI scripts, to the most sophisticated Enterprise Application Servers. Its simple HTTP based interface and pre-built toolkits can be used by nearly any server-side web programming language (e.g. .NET, JSP, Java, CFML, PERL and C) to dynamically create chart enabled web pages.

NetCharts Performance Dashboards

NetCharts Performance Dashboards (http://www.visualmining.com/ncpd/) is a complete end-user enabling, agile, dashboarding solution. NetCharts Performance Dashboards allows users or end-users to view, create and customize dashboards. No coding or programming is required and there are no languages to learn to connect to your data and produce and present dashboards. It's all done automatically based on selections, allowing for rapid, agile dashboard development with rich client-side interactivity.

Chart Definition Language

Visual Mining's *Chart Definition Language (CDL)* is a simple ASCII scripting language. There are 20 basic chart types defined in CDL, each of which has hundreds of configurable attributes, allowing for the creation of thousands of different charts. See the *Visual Mining Chart Definition Language Reference Guide* for a complete description of CDL.

```
ChartType = LINECHART;
ChartName = "Basic Line Chart";
ChartSize = (400,250);
Background = (, BOX, 1, , TILE, silver, SQUARE, SQUARE, SQUARE, SQUARE, );
Header = ("Linechart", grey, "Arial", 16, 0, CENTER, CENTER, "OFF");
GridLine = (BOTH, SOLID, 1);
Grid = (xc0c0c0_89, white, white, , TILE);
LineSets = ("LineSet1", );
LineSet1 = 100, 125, 245.78, 147, 67;
LineDropShadow = (x000000_110, 0.01, 0.0090, 0.02);
LineSymbolSpotlights = (xffffff_150, xffffff_0, CENTER, 0, 0, -0.5, -0.25,
0.9999);
LineSymbol = (CIRCLE, 9, FILLED, , 1, , blue, 0);
LineStyle = (SOLID, 1, blue, , NORMAL, );
LeftMajorTicColor = x000000_0;
LeftColor = x000000_0;
LeftScale = (0, 300, 50);
LeftTics = ("", grey, "Arial Plain", 11, 0, CENTER, , CENTER);
BottomLabels = "January", "February", "March", "April", "May";
BottomMargins = (8, 13);
BottomTics = ("", grey, "Arial Plain", 11, 0, CENTER, , CENTER);
BottomMajorTicColor = x000000_0;
BottomColor = x000000_0;
```

Figure 1-CDL for a Linechart

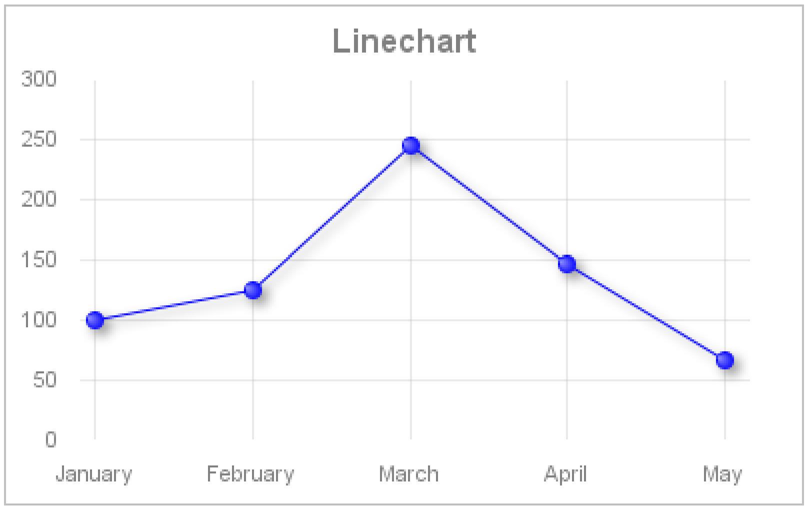
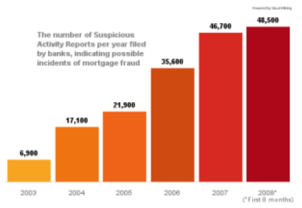


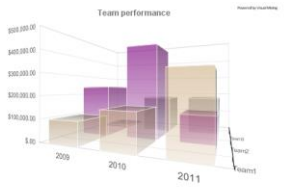
Figure 2 - Linechart created by CDL in Figure 3

NetCharts Pro Chart Types

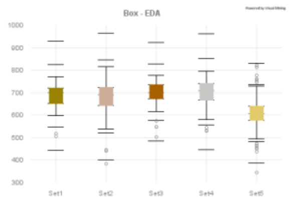
All of Visual Mining's charting solutions use the Chart Definition Language (CDL) to create and manipulate charts. This common use of CDL makes it easy to preserve chart definitions when moving from one product to another. The following shows the 20 chart types that are supported as Java Applets with NetCharts Pro:



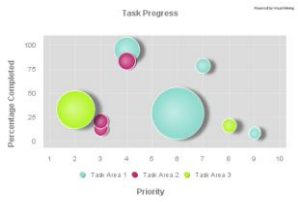
Bar



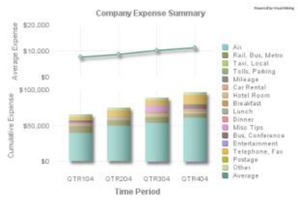
3d bar



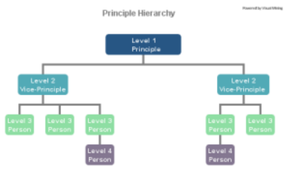
Box



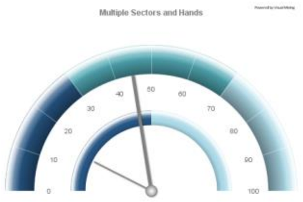
Bubble



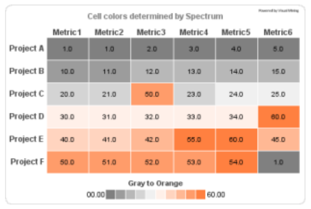
Combo



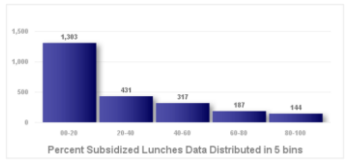
Diagram



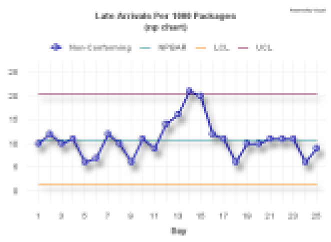
Dial



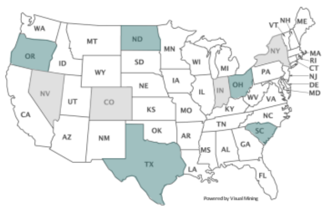
Heat map



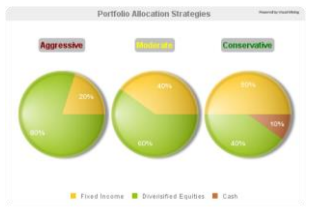
Histogram



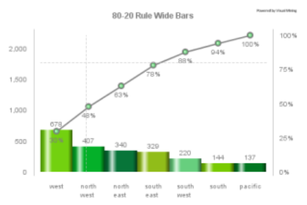
Line

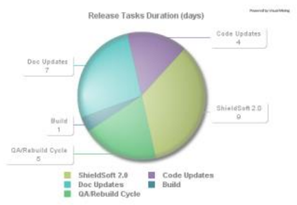


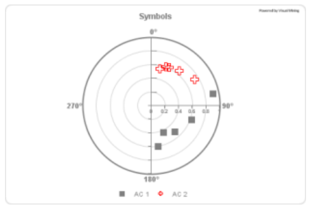
Map



Multi-pie

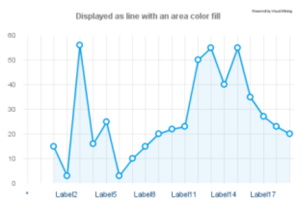

Pareto

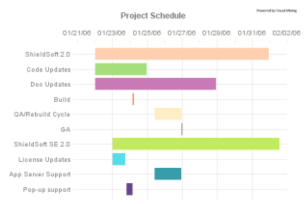

Pie

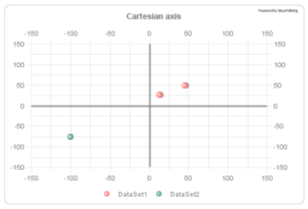

Polar

Radar

Stock


Strip


Time


XY plot

3.0 NetCharts Pro Feature Set

NetCharts Pro provides a programmer-friendly interface to the NetCharts charting engines. In addition to exposing NetCharts features, NetCharts Pro provides image generation capabilities, classes for modeling chart data, and a robust Java API.

Image and ImageMap Generation

NetCharts Pro is capable of producing images and HTML image maps of charts. This feature makes NetCharts Pro ideally suited for use in servlets and JSP pages that need to incorporate interactive chart images into web pages. The mime types in the following list represent the image types supported by NetCharts Pro.

- image/png
- image/svg+xml
- image/cmu-raster
- image/bmp
- image/jpg
- image/pcx
- image/pict
- image/psd
- image/tga
- image/tiff
- image/xbm
- image/xpm
- image/wbmp

Raster Image Formats

The PNG (`image/png`) format is best suited for web applications. It produces small, high quality, non-proprietary files supported by all modern browsers.

The image maps are built using data from the `ActiveLabel` parameters in the chart definition. Nearly every item on a chart, including titles, axes, legends and individual data points has an associated `ActiveLabel`. NetCharts Pro provides convenience methods that automatically generate standard MAP tags. It also provides methods that return objects containing all information necessary for programmers to construct specialized map tags. See the Javadoc documentation for the `netcharts.pro.util` package for detailed information on the alternatives for constructing image map tags.

Vector Image Formats

The SVG (`image/svg+xml`) mime type produces charts in Scalable Vector Graphics (SVG) format which is compatible with the HTML 5 specification. This format allows rich client side interaction beyond what is available in the other bitmap image formats. Build animations allow chart elements to grow or fade into place. Legends can be made interactive, allowing users to temporarily hide/show data series by clicking on the associated legend item.

SVG is NetCharts Pro's default vector format. NetCharts Pro can also produce SWF or Flash output if NetCharts Pro determines the hosting browser is not modern enough to support native SVG objects. NetCharts Pro includes javascript libraries that will automatically translate SVG into Flash after the chart has been delivered to the end user's browser.

Federal IT Accessibility Initiative Compliance

NetCharts Pro contains features that allow it to be used in U.S. Government sponsored applications that are required to provide web pages that are accessible to people with disabilities. The details of this Federal IT Accessibility Initiative can be found at http://www.section508.gov.

NetCharts Pro contains an API call to allow programmers to get text suitable for use as the ALT attribute of an HTML IMG tag. This textual description can be explicitly defined by the chart author, or automatically generated by NetCharts Pro.

Swing Based Graphics

NetCharts Pro charts are lightweight components based on the Java Swing Toolkit. Java based desktop applications can use NetCharts Pro for visualizing charts within Java applications.

Java Applets

NetCharts Pro supports rendering interactive visualizations as Java applets. In applets Mode, NetCharts Pro can allow developers of chart-enabled applications using any web scripting language such as ASP, ASP.NET, JSP, PHP, etc. to include interactive charts.

Recent browser updates are changing the way unsigned Java applets and web start applications are run. The default security level for Java applets and web start applications has been increased from "Medium" to "High" affecting the conditions under which unsigned Java web applications can run. Previously, as long as you had the latest secure Java release installed applets and web start applications would continue to run as always. With the "High" setting the user is always warned and requires that users indicate acceptance before any application is run and the JRE may not allow unsigned applications to run at all, to prevent silent exploitation. A signed JAR file is now recommended for applet deployments.

A signed version of the NetCharts Pro core library, **ncp-core.jar,** used to serve NetCharts Pro charts in applets mode is available within the NetCharts Pro distribution in the /**applets** folder. Using this signed JAR file and running your applets in sandbox mode, it is possible to limit the warning dialogs to a single dialog when the first NetCharts Pro is presented.

To run the applet in **sandbox** mode, include a permissions element inside the applet element like:

```
<applet id='netcharts' codebase='/netcharts' archive='ncp-core.jar'
code='netcharts.apps.NFBarchartApp' width='400' height='400'>
      <param name='permissions' value='sandbox'/>
      <param name='NFParamScript' value='
            DataSets=("BarSet1",null,BAR,4,FILLED);
            DataSet1=100,125,245.78,147,167,120,60;
            BarLabels="Mon","Tue","Wed","Thu","Fri","Sat","Sun";'/>
</applet>
```

If you are an existing user and accessing templates or CDL files on a different port or server using the *NFParamURL* feature, use the signed JAR file within the **/applets/all-permissions** folder to allow cross-domain access. To run the applet with **all-permissions**, include a permissions element inside the applet element like:

```
<applet id='netcharts' codebase='/netcharts' archive='ncp-core.jar'
code='netcharts.apps.NFBarchartApp' width='400' height='400'>
      <param name='permissions' value="all-permissions'/>
      <param name='NFParamURL' value="bar.cdl'/>
</applet>
```

See *Web Scripting with NetCharts Pro Applets* for more details on NetCharts Pro Applets.

Data Models

NetCharts Pro provides a set of data modeling interfaces that simplify the process of programmatically specifying the data to be loaded into a chart. Programmers can write classes that implement a NetCharts Pro data model interface to extract chart data from arrays, JDBC result sets, DOM objects or any other data structure. Data model interfaces are provided for one, two and three-dimensional data. See the JavaDoc documentation for the `netcharts.pro.datamodel` package for more detailed information on the data models. The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including many with sample DataModel implementations that can be used for access to existing data or as a basis for your own DataModel implementation.

Chart Events

NetCharts Pro allows other components in an application to register as ChartActionEvent listeners. Such listeners will be informed through a ChartActionEvent when the mouse rolls-over or is clicked over any chart item (e.g., legend items, pie slices, bar values, titles, etc...). ChartActionEvents also report scrolling, zooming and repaint events. See `SimpleEventExample.java` in the examples folder of the NetCharts Pro source distribution for an example using this feature.

4.0 NetCharts Pro APIs

NetCharts Pro offers a comprehensive object-oriented API for every aspect of the chart. The object-oriented API provides a rich set of objects that can be used to control every attribute of a chart. The character string based API consists of a small set of methods that deal with strings of CDL. The string based API is less intuitive than the object API, but provides higher performance. Care should be taken to use one API or the other within a single program. Mixing the API's in a single program can produce unpredictable results.

Object-Oriented API

The object-oriented API consists of hundreds of objects and their accessor methods. This API is immediately familiar to Java programmers. The following code sets the header of a chart

```
NFTitle header = chart.getHeader();
header.setText("Object-based API Header");
Font headerFont = new Font("Helvetica",Font.PLAIN, 16);
header.setFont(headerFont);
header.setTextColor(Color.black);
chart.setHeader(header);
```

See the sample code in the NetCharts Pro distribution for detailed examples of the use of the object-oriented API. Most modern Java Integrated Development Environments produce command-completion and in-line help based on this API. This API is particularly useful when building applications that contain chart editors. Graphical editor panels representing the state of certain chart elements can be implemented based on the associated chart objects.

String-based API

The string-based API consists of six `set()` methods located on netcharts.pro.common.NFGraph, the abstract superclass for all NetCharts Pro charts. These six methods allow for older NetCharts Beans users to easily migrate to NetCharts Pro and have an implicit dependence on the syntax of CDL. The following code sets the header of a chart. It produces the same result as the object oriented code presented above.

```
      chart.set("Header =(\"String-based API Header\",black,\"Helvetica
Plain\",16,0);");
```

The other `set()` method signatures allow modification of individual attributes of individual CDL parameters. See the javadoc API documentation of the `set()` methods for complete details.

The string-based API produces more compact code than the Object Oriented API. The string-based API is more efficient and provides better performance - it avoids the creation of a number of Java objects and eliminates processing by the rendering engine to extract the chart attributes from those objects.

5.0 NetCharts Pro Resource Utility Servlet

NetCharts Pro includes a utility servlet that *must* be present in a NetCharts Pro enabled web application. The servlet is `netcharts.pro.util.NFResourceServlet` and it is delivered in **ncp-api.jar**. This servlet is responsible for delivering a variety of content to web applications including binary image data, vector image files, javascript libraries, applet jars and license information. This servlet can be integrated into a web application framework and configured with the following entries in the `web.xml` deployment descriptor.

```
<servlet>
    <servlet-name>NCProResourceServlet</servlet-name>
    <display-name>NCProResourceServlet</display-name>
    <servlet-class>netcharts.pro.util.NFResourceServlet</servlet-class>
      <init-param>
            <param-name>RetainImageInSession</param-name>
            <param-value>true</param-value>
      </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
<servlet-mapping>
      <servlet-name>NCProResourceServlet</servlet-name>
      <url-pattern>/getresource</url-pattern>
</servlet-mapping>
<servlet-mapping>
      <servlet-name>NCProResourceServlet</servlet-name>
      <url-pattern>/NFLicense.dat</url-pattern>
</servlet-mapping>
<servlet-mapping>
      <servlet-name>NCProResourceServlet</servlet-name>
      <url-pattern>/netcharts.jar</url-pattern>
</servlet-mapping>
```

Content Delivery

`NFResourceServlet` is responsible for delivering NetCharts Pro generated content to web pages. HTML code generated by the NetCharts Pro chart and page creation code will contain indirect references to this servlet in order to request content.

For example, an NetCharts Pro application might issue a call to create an HTML page containing a pie chart:

```
String page = NFServletUtil.getPage(request, chart, new
NFRasterPageParams(NFRasterPageParams.MIME_TYPE_PNG));
```

This call results in the creation of an HTML page containing code that looks like

```
<img src="getresource?image=1611207186" border="0" style="z-index:0;"/>
```

Given the above `web.xml` entries, `getresource` is a reference to `NFResourceServlet` which, in this case, retrieves binary PNG image data and returns it to the browser. Other HTML code created by NetCharts Pro may make similar requests to `NFResourceServlet` for other types of content.

`NFResourceServlet` supports a `RetainImageInSession` initialization parameter. Setting `RetainImageInSession` to true will ensure that once a chart image has been generated, it will be available for re-delivery in response to subsequent requests until the session times out, or until the image is explicitly deleted from the session. This is useful in the event that the user re-requests the image either by reloading the web page, or by printing the web page. The servlet's memory usage will continue to grow as each unique chart is generated and stored in session. Explicit session management may be required to prevent excessive memory consumption.

`NFResourceServlet` supports an `AlternateFileLocation` initialization parameter. This instructs the servlet to look in alternate locations for files. For example, a developer can provide updated versions of certain javascript libraries used by NetCharts Pro by using `AlternateFileLocation` to point to a directory containing the preferred version of those libraries.

License Delivery

`NFResourceServlet` is also responsible for checking that an application contains a valid NetCharts Pro license. By default, the servlet expects the license to be located in **WEB-INF/classes/NetChartsPro.license**.

The `NFResourceServlet` servlet does support initialization parameters that alter the way it locates a license. The `LicenseDirectory` parameter can be used to specify a directory within the web application other than **WEB-INF/classes** in which to search for a license. The `License` parameter can contain a base 64 encoded string version of the contents of **NetChartsPro.license**, in which case a license file does not explicitly need to be present in the web application. Note the use of these two parameters require that the servlet be loaded at startup as shown above.

6.0 Building Programs with NetCharts Pro

NetCharts Pro can be used to chart-enable a variety of Java programs including desktop applications, applets, chart image generation servlets and JSP pages. This chapter presents general guidelines for chart enabling various types of programs using NetCharts Pro 7.2.

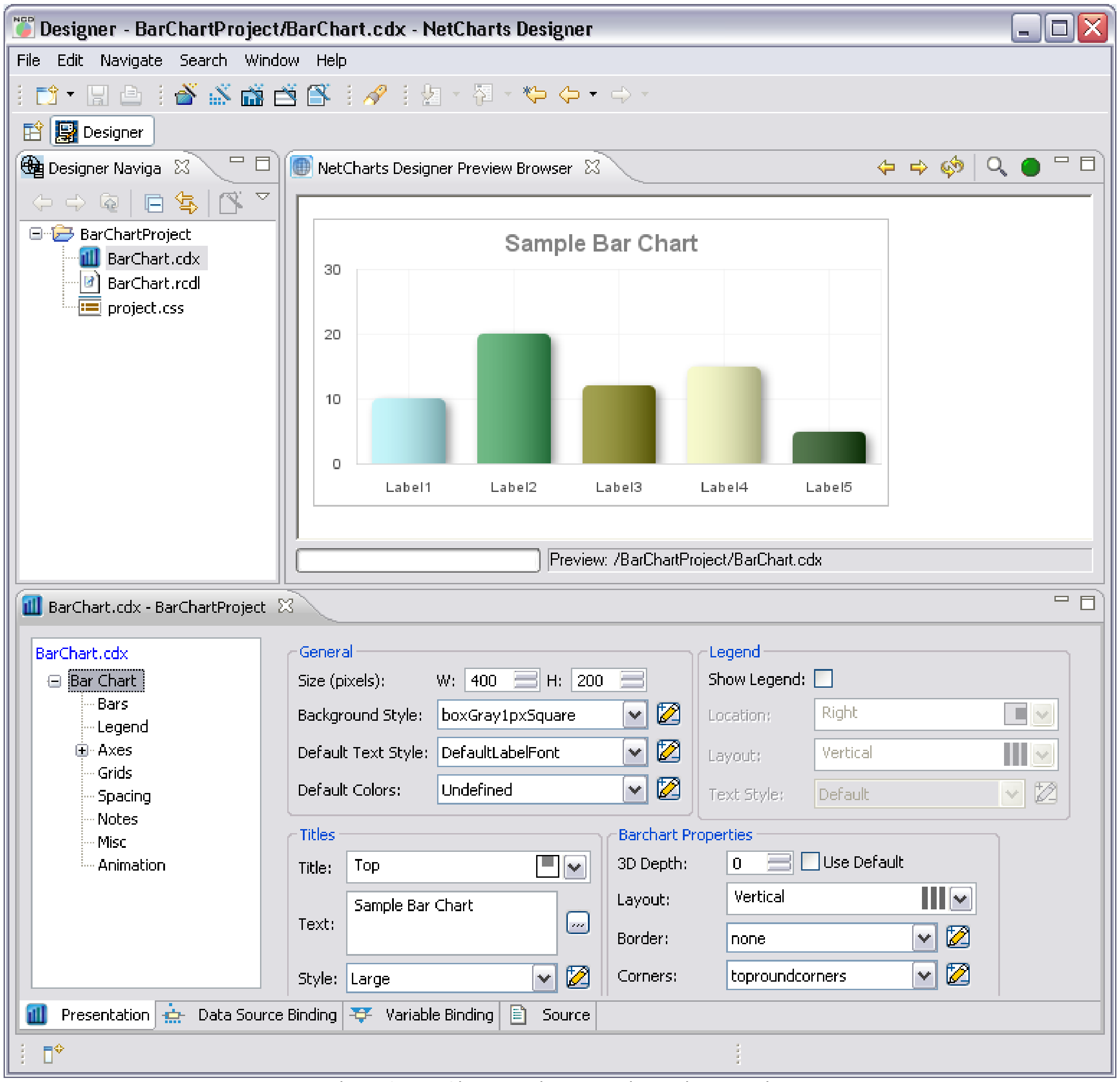
Creating Chart Templates

It can be useful to consider using *chart templates* when designing applications that use NetCharts Pro. Chart templates are files or strings of CDL that define a chart or portions of a chart. Chart templates support the notion of separating a chart definition into static and dynamic components. The static component of a chart definition contains all those attributes that do not change between invocations. The dynamic component of a chart definition contains those attributes that are unique to a particular chart request.

Consider an application that contains a bar chart. Much of the definition of that bar chart will likely be static. For instance, its background may always be white; it may always use 12 point Helvetica fonts; and it may contain a title, a legend and two axes. The dynamic component of the chart might be limited to the actual bar values, the axis labels and the text of the title.

The static components of charts can be created in advance and stored in chart templates. At runtime, the application could create an instance of a NetCharts Pro chart, initialize it with a chart template, add any additional dynamic data using appropriate API calls, and then generate the chart image.

Creation of chart templates can be done inside the application code, or externally with NetCharts Designer. NetCharts Designer is a desktop design tool that allows chart authors to visually design chart templates. NetCharts Designer provides graphical editors to allow quick logical access to every chart attribute. The resulting chart definition can be exported as a file of Resolved CDL (RCDL) and used in the construction or initialization of a NetCharts Pro object. Resolved CDL has had all references to styles, and external data resolved into CDL parameters. Figure 4 on the following page shows Designer being used to create a chart template.



General
Size (pixels): W: 400 H: 200
Background Style: boxGray1pxSquare
Default Text Style: DefaultLabelFont
Default Colors: Undefined

Legend
Show Legend: ☐
Location: Right
Layout: Vertical
Text Style: Default

Titles
Title: Top
Text: Sample Bar Chart
Style: Large

Barchart Properties
3D Depth: 0 ☐ Use Default
Layout: Vertical
Border: none
Corners: toproundcorners

Figure 3 - NetCharts Designer creating a chart template

Creating and using chart templates also simplifies the process of migrating a NetCharts Pro application to one of Visual Mining's other charting solutions. A chart template created for use with NetCharts Pro can also be used by NetCharts Server, allowing programmers to adapt their programs to changing application infrastructures.

A General Structure for Chart-enabled Programs

Regardless of the type of program being developed, there is a simple structure that can be used to integrate charts using NetCharts Pro. This structure contains the following elements:

- Preliminaries - Setting the license key
- Instantiating a chart
- Loading the static component of the chart definition
- Loading the dynamic component of the chart definition

- Configuring chart event listeners (desktop applications)
- Displaying or rendering the chart

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications that follow this structure.

Preliminaries - Setting the License Keys

A NetCharts Pro enabled application must have access to its license at runtime. An evaluation version of the required license is contained in the product distribution as a file called **NetChartsPro.license**. Activated licenses are distributed as a separate file at the time of purchase or renewal.

Setting the license can be accomplished in two ways. The file containing the license can be placed in the program's CLASSPATH, or the license can be set with an API call within the program.

The most common See Chapter 5 for details on how the NFResourceServlet is used to perform license checks using the CLASSPATH.

A second way to provide the chart with its license information is to set the license programmatically. This eliminates the configuration issues associated with discovering licenses at runtime.

```
static {
      netcharts.pro.common.NFGraph.setLicenseKey("XXX…");
}
```

The string that is used as an argument for this API call must be explicitly generated using the content of a valid **NetChartsPro.license**. The NetCharts Pro distribution contains a utility program that will translate the content of a license file into a string that can be used by this API.

Instantiating a chart

NetCharts Pro provides a set of static chart creation methods defined in the NFGraph superclass. These static creators can create charts from templates represented as File objects, Input Stream objects, URL's or strings of CDL.

```
public static NFGraph getGraphFromTemplate(InputStream templateStream)
public static NFGraph getGraphFromTemplate(File templateFile)
public static NFGraph getGraphFromTemplate(URL templateURL)
public static NFGraph getGraphFromTemplate(String template)
```

These static creators examine the given chart template and create and return the appropriate type of chart instance.

Other variants of these static creators take an additional `Properties` object as an input argument. This allows setting runtime properties for a chart. The following table shows the available runtime properties that can be passed to NetCharts Pro chart creation methods.

MaxDataSets	Sets the maximum number of data sets a chart can support. The default is 50.
MaxAxes	Sets the maximum number of Axes a chart can support in a single dimension. The default is 10.
DateFormats	Define legal date strings formats in a chart. The default is the standard Java SimpleDate format.

The following sample code demonstrates setting runtime properties as part of chart instantiation.

```
Properties props = new Properties();
props.put("MaxDataSets", "75");
NFGraph g = NFGraph.getGraphFromTemplate(template, null, props);
```

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications with full source code that demonstrate these creator methods.

Loading the static component of the chart definition

The static components of a chart definition can be loaded into a chart in several ways. They can be provided at creation time, or they can be added to an empty chart instance with a series of calls to the various API set methods (`setHeader()`, `setBackgroundRegion()`, etc).

Loading the dynamic component of the chart definition

Once the chart has been created and populated with the static portions of the chart definition, it can be populated with instance specific data. Typically, the program will interrogate some data source to obtain the appropriate data and add it to the chart. Data can be added directly using the chart specific set methods. More complex applications can create and populate data models using the `netcharts.pro.datamodel` package.

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including an application called *Displaying charts in Servlets*, implemented primarily as the servlet, `SimpleServlet`, which demonstrates using NetCharts Pro to present an interactive chart from a Servlet and Loading data into a project schedule which demonstrates a sample data model implementation. The full source code of the example is available from the demonstration application.

Configuring chart event listeners

Desktop programs containing NetCharts Pro can observe and interact with charts at runtime. The `NFChartActionListener` interface allows programs to monitor mouse activity over a chart. ChartActionEvents describe mouse dwell events and mouse click events over individual chart components. Mouse activity over chart data points, labels, titles, legends, and axes generates descriptive events that can be captured and acted upon by the parent program. See `SimpleEventExample.java` in the examples folder of the NetCharts Pro source distribution for an example using this feature.

Displaying or rendering the chart

Programs containing NetCharts Pro will ultimately either draw a chart on the screen, or create an image of a chart to be delivered to another program to be displayed. Drawing a chart on the screen involves placing the chart into the desired location in the component hierarchy of the parent program.
For example:

```
// create the Frame that will contain the chart.
JFrame f = new JFrame("Visual Mining NetCharts Pro Barchart Example");
f.getContentPane().setLayout(new BorderLayout());
f.getContentPane().add(chart.getPanel());
f.setSize(chart.getPanel().getSize());
f.show();
```

In server side programs, such as servlets, the chart is not drawn to the screen; rather it is drawn in memory and then used as the source to create an image in a particular image format (e.g. PNG or JPG). In this case it is not necessary to create a JFrame or any other graphical component.

```
      // NFServletUtil.getPage processes the chart and generates the HTML to
include on the web page.

      String theChartHtml = NFServletUtil.getPage(request, chart, new
NFRasterPageParams());
```

NetCharts Pro can also return just the rendered chart image data in an `NFServerGeneratedImage` object. This object contains the encoded image as a byte array. Programmers can write this byte array to disk, a session variable or stream it back as a response to an HTTP request.

```
      // NFImageGeneration.generateImage processes the chart and generates an
NFServerGeneratedImage with the rendered image as a byte array.

      NFServerGeneratedImage theChartImage = NFImageGeneration.generateImage
(chart, new NFPNGImageParams());
```

Retrieving a generated image map

For web applications, NetCharts Pro can also deliver image maps to accompany the chart images it produces. Image maps complement chart images with the ability to display hover and rollover labels and the ability to drill through to other information. NetCharts Pro provides several mechanisms for creating image maps.

The n`etchart.pro.util.NFServletUtil` convenience class provides the simplest way to create image maps. It contains a static method, `getPage()`, that completely automates the creation of an HTML page containing a chart image and related image map.

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including an application called *Displaying charts in Servlets*, implemented primarily as the servlet, `SimpleServlet`, which demonstrates using the `getPage()` API method to present an interactive chart in a web application.

If more control is needed to override NetCharts Pro's automatic image and image map generation, convenience classes are offered for each step of the process. The `NFServerGeneratedImage` object contains data structures necessary to manually create an image. The `NFImageMapCreator` class is a higher level convenience class that simplifies the process of building an image map from an `NFServerGeneratedImage` object.

Client-side interactivity

A useful way to increase the user-friendliness of generated charts is to introduce client-side interactivity to the displayed web page. One way to produce the interactivity is through JavaScript. NetCharts Pro offers multiple out-of-the box user interactivity capabilities including hovering "rollover" labels, legend highlighting of entire bar sets, lines or slices and the ability to turn off sections of a chart by selecting the legend item representing the data set.

Additionally, custom JavaScript functions can be called to perform business logic or additional tasks when a user selects an area of the chart. These actions can be configured using the associated PageParams when calling the `getPage()` API.

In a production environment, the JavaScript may be stored in a database, templates within a directory structure, a web service or other locations. In addition, the JavaScript may be generated much like the chart images, with dynamic data based upon the query parameters.

Debugging NetCharts Pro programs

The utility class `NFDebug` can be used to turn on various levels of debug output. See the API documentation for details on each option. Adding the following line to a NetCharts Pro program will print detailed diagnostics as the chart starts and runs.

```
        netcharts.util.NFDebug.set("ALL");
```

While debugging it can also be useful to extract and examine the complete chart definition that the chart is using. This can be done with the following command:

```
        System.out.println(graph.toString());
```

Deploying NetCharts Pro programs on Unix Platforms

Visual Mining's products are written in Java and contain code that creates graphical objects (charts). This can introduce a dependency on the host system's native graphics environment. On UNIX platforms, Java's graphics libraries (the Java AWT) are implemented on top of XWindows/Motif graphics libraries and by default depend on access to an XServer in order to obtain font metrics and other graphics information.

NetCharts Pro is commonly run on UNIX machines that do not have XServers. Java Servlets that do server-side chart image generation are often deployed on production machines that are headless (i.e. a machine without a monitor/mouse/keyboard attached) and do not have a physical XServer installed.

Sun introduced a headless processing mode in starting with JVM 1.4. Running NetCharts Pro 7.2 with a JVM 1.6 or greater using the `-Djava.awt.headless=true` flag eliminates the need for an XServer.

7.0 A Sample Application

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including an interactive application called *Showing detailed information via drilldown*, implemented primarily as the servlet, `DrillDownExample`, which demonstrates using NetCharts Pro to create an interactive, multi-layered chart. This example servlet creates a NetCharts Pro `NFGraph` instance based upon a default template. Then the servlet uses query string parameters to modify the chart instance. Once the chart has been dynamically modified, the chart instance is then used to invoke the NetCharts Pro `getPage()` method to generate the chart image which is returned to the client.

In addition, the sample application shows additional techniques to integrate NetCharts Pro within an interactive environment. The sample application details how to add additional functionality such as rollover labels using HTML imagemaps, customizing a chart with dynamic chart properties, including additional chart properties across a series of charts and creating additional chart layers using the same servlet. Web applications such as the `DrillDownExample` Servlet can either directly return an image or use a Model-II architecture in which the image is placed in memory by the servlet and later accessed by another page (discussed in Chapter 8: Web Programming using NetCharts Pro).

Supported Parameters

The `DrillDownExample` servlet uses two sets of query string parameters. The first set is used primarily for navigation and the second set is used for the dynamic chart template creation. The following query string parameters are accepted as input:

`image`

used for servlet navigation. Instructs the servlet to return the binary image data directly and is useful when using the servlet in an IMG SRC tag.

`goback`

used for servlet navigation. Instructs the servlet to return one level upwards within the multi-layered data.

`year`

used for dynamic chart template creation. Specifies the year of the data set to represent.

`region`

used for dynamic chart template creation. Specifies the region of the data set to represent.

`city`

used for dynamic chart template creation. Specifies the city of the data set to represent.

`category`

used for dynamic chart template creation. Specifies the category of the data set to represent.

Control Flow

Figure 4 depicts the interaction between the servlet within the sample application and NetCharts Pro. After NetCharts Pro and the sample application have been deployed within the servlet container, a web browser, web page or other client (which in this example is the servlet itself) can dispatch a URL that will cause the application server to invoke the servlet and generate a chart.

In the sample application, the initial reference to the `DrillDownExample` servlet is forwarded to the page `ncpdrilldown.jsp` page. This is to ensure compatibility across multiple application servers. Any future invocations of the servlet come from with the tags returned by the servlet.

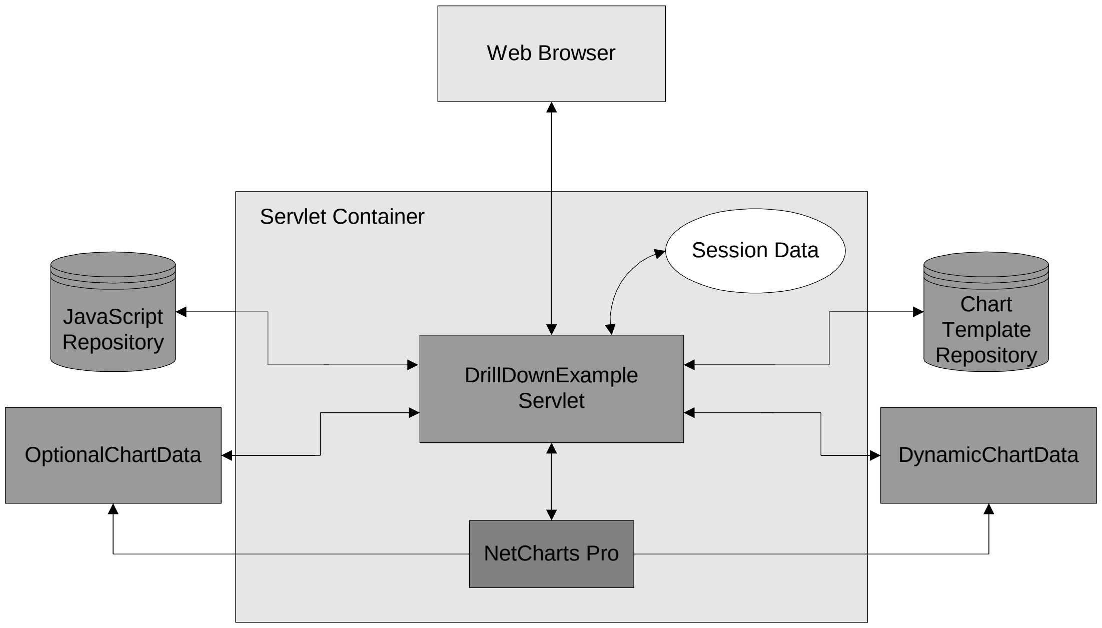


Figure 4- Netcharts Pro and the sample application deployed in a servlet container

As an example, consider the following invocation of `DrillDownExample`, made by the servlet during navigation, which requests a chart image representing the sales data where the year is 2000 and the region is East (result shown in Figure 5).

```
http://<host>:<port>/<webapp_name>/ncpdrilldown?year=2000&region=east
```

The specific steps that occur during this request are as follows:

1. The browser or calling client dispatches the URL.
2. The servlet container (Tomcat, Application Server, etc.) launches the servlet.
3. The servlet retrieves a chart template from the Chart Template Repository. In this example, the Chart Template Repository is a Java class that simply returns a static chart template.
4. The servlet creates an NFGraph instance of the NetCharts Pro using the initial Chart Template.
5. The servlet creates an instance of DynamicChartData. The DynamicChartData instance will generate dynamic chart data based upon the query parameters and set the appropriate properties of the NFGraph instance. The DynamicChartData class shows a classic example of modifying the properties of a chart directly.
6. The servlet creates an instance of OptionalChartData. The OptionalChartData instance will generate additional chart data and set the appropriate properties of the NFGraph instance. The OptionalChartData class shows a classic example of modifying the properties of a chart directly.
7. The servlet instructs the chart to redraw itself using all the newly defined parameters.
8. The servlet renders the chart and associated image map data using the NFServletUtil convenience class and adds the content as a request attribute for the forwarding page to consume. The getPage API stores the NFServerGeneratedImage into a session variable.

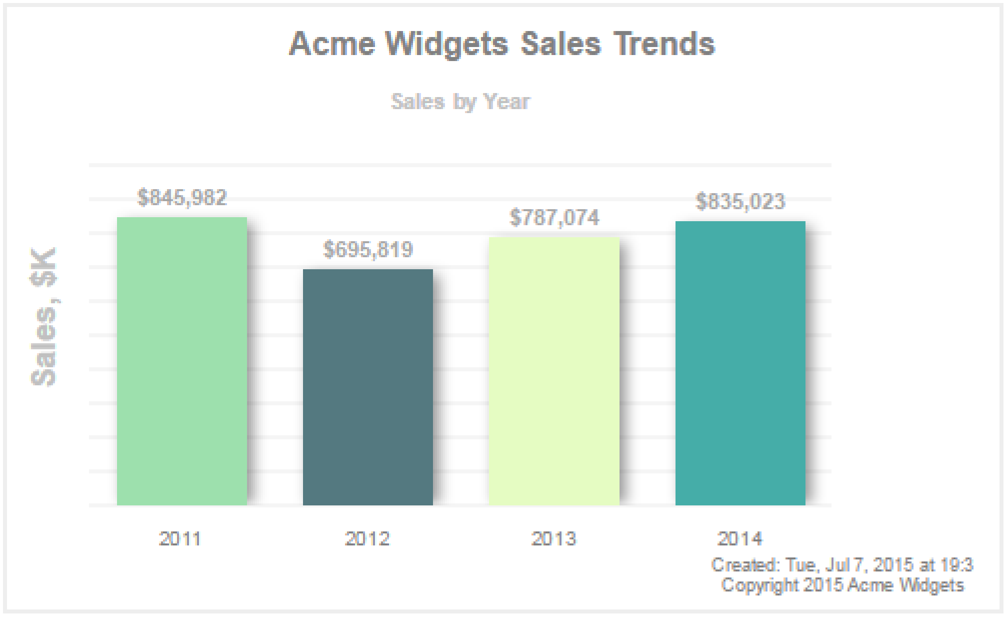


Figure 5- Example generated chart and image map from Sample Application

The fully annotated source code for the sample application, including the servlet and helper classes, can be found in the NetCharts Pro Examples at http://ncpro.visualmining.com/ncprowebexamples/.

8.0 Web Programming using NetCharts Pro

When returning data to a requesting web browser, a Java Servlet can only return one type of data in a response (for example either text or binary). Generally servlets return an HTML page in response to a request. A servlet using NetCharts Pro to create charts images cannot return both an HTML page and the binary chart image in the same response because of the single return type limitation of servlets and the fact that the MIME types of the image (`img/png`, `img/jpg`, etc.) are different than HTML (`text/html`). NetCharts Pro overcomes this limitation by providing a built in utility servlet that handles the routine delivery of all content necessary to produce a web page containing interactive charts.

Recommended Application Architecture

Figure 7 shows the recommended architecture for NetCharts Pro enabled web applications. Such a web application will contain one or more chart generation servlets or chart generation JSP files. The application also contains NetCharts Pro's NFResource Servlet. The process flow for this architecture is as follows:

1. An end user launches a web application.
2. The web application contacts the application server and executes a chart generation servlet or JSP page.
3. The chart generation servlet makes an API call to NetCharts Pro to produce a chart.
4. NetCharts Pro produces a chart, writes the chart image data to session and creates a fragment of HTML that will request the image by way of the NFResourceServlet.
5. The chart generation servlet returns the HTML produced by NetCharts Pro to the browser.
6. The browser executes the HTML code produced by NetCharts Pro, which results in a request to the NFResource servlet for the chart content.
7. The NFResource servlet returns the chart content.

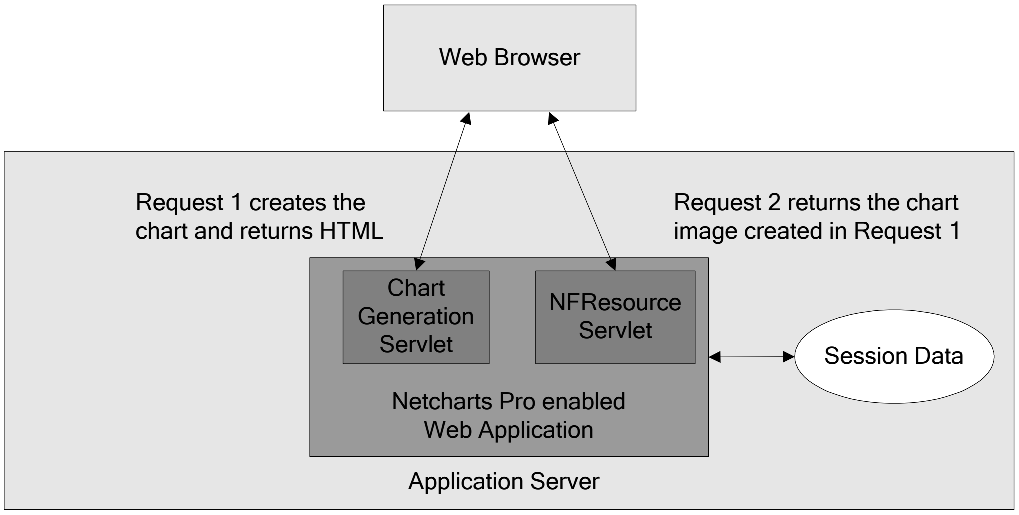

Web Browser
Request 1 creates the chart and returns HTML
Request 2 returns the chart image created in Request 1
Chart Generation Servlet
NFResource Servlet
Session Data
Netcharts Pro enabled Web Application
Application Server

Figure 6 NetCharts Pro-enabled servlet architecture

The Chart Generation servlet utilizes the NetCharts Pro `netcharts.pro.util.NFServletUtil` convenience class which contains methods to assist with adding charts, both standard and interactive, to web pages. The primary methods of the `NFServletUtil` class are `getPage` and `getScript`

- The `NFServletUtil.getPage` method will return a `java.lang.String` containing the HTML `img` or object element, JavaScript and any HTML `map` and `area` elements if needed, to add the chart to the page. To add just a chart image to a page without an image map, call the API `setMapIncluded(false)` on the `NFImageParams` instance used to define the PageParams parameter of the the `getPage` method signatures. The call to `getPage` will produce HTML that contains a reference to the NFResource servlet, which will be called to fetch the image data.
- The `NFServletUtil.getScript()` method will return a `java.lang.String` containing the JavaScript code used for client-side interactivity. This interactivity is also referred to as "Active Labels". Adding the returned String to the page will ensure that all interactive charts included within the page have access to the code needed for client-side interactivity.

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including an application called *Displaying charts in Servlets*, implemented primarily as the servlet, `SimpleServlet`, which demonstrates using NetCharts Pro to present an interactive chart from a Servlet. The full source code of the example is available from the demonstration application.

JSP pages function using this same architecture. The server-side source code for the JSP page contains the API calls to NetCharts Pro to create the charts and the HTML. The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including an application called *Displaying charts in JSP pages*, implemented primarily as the JSP page, `ncpjspexample.jsp`, which demonstrates using NetCharts Pro to present an interactive chart from a JSP page. The full source code of the example is available from the demonstration application.

Chart Output Type Options

NetCharts Pro can deliver charts in bitmap or vector output formats. Bitmap images, such as PNG or JPG are lightweight, fast to deliver and display in a browser and easily printed or repurposed in external documents. Vector formats such as SVG offer more interactivity features, such as build animation, zoom and dynamic display/undisplay of individual data series.

Generating charts as PNG or JPG

NetCharts Pro contains convenience utilities that simply the process of generating a bitmap chart image and the HTML that can be used to display that image. In most cases a method from the `NFServletUtil` class can be used to produce the desired chart with the desired bitmap image type – typically PNG or JPEG.

```
      String theChartHtml = NFServletUtil.getPage(request, chart,
new NFRasterPageParams(NFRasterPageParams.MIME_TYPE_PNG));
```

This code also produces an HTML image map tag associated with the image that automatically supports rollover and drilldown functions.

Generating charts as SVG

NetCharts Pro contains convenience utilities that simply the process of generating a vector chart image and the HTML that can be used to display that image. The default vector image type is SVG.

```
      String theChartHtml = NFServletUtil.getPage(request, chart,
new NFSVGPageParams());
```

This code produces an HTML page that contains inlined SVG representation of the requested chart. Such HTML code can be rendered in any HTML 5 compliant web browser.

Vector support for older browsers

For older, non HTML5 compliant browsers, it is still recommended to present charts as PNG images. But NetCharts Pro supports vector output to older non HTML 5 compliant browsers by translating its SVG output into the Adobe Flash vector format. This translation occurs via JavaScript in the end user's browser after the SVG chart has been delivered. NetCharts Pro includes a JavaScript library that will translate SVG to Flash. This allows developers to develop HTML 5 compliant web applications that can function equivalently in any modern web browser.   The following code demonstrates how to activate this behavior.

```
NFSVGPageParams svgOptions = new NFSVGPageParams();
svgOptions.setSVGWebIncluded(true);
svgOptions.setSVGWebForced(false);
theChartHtml = NFServletUtil.getPage(request, chart, svgOptions);
```

This code causes the generated HTML to include directives to include in the page for JavaScript libraries that translate SVG into Flash as needed.

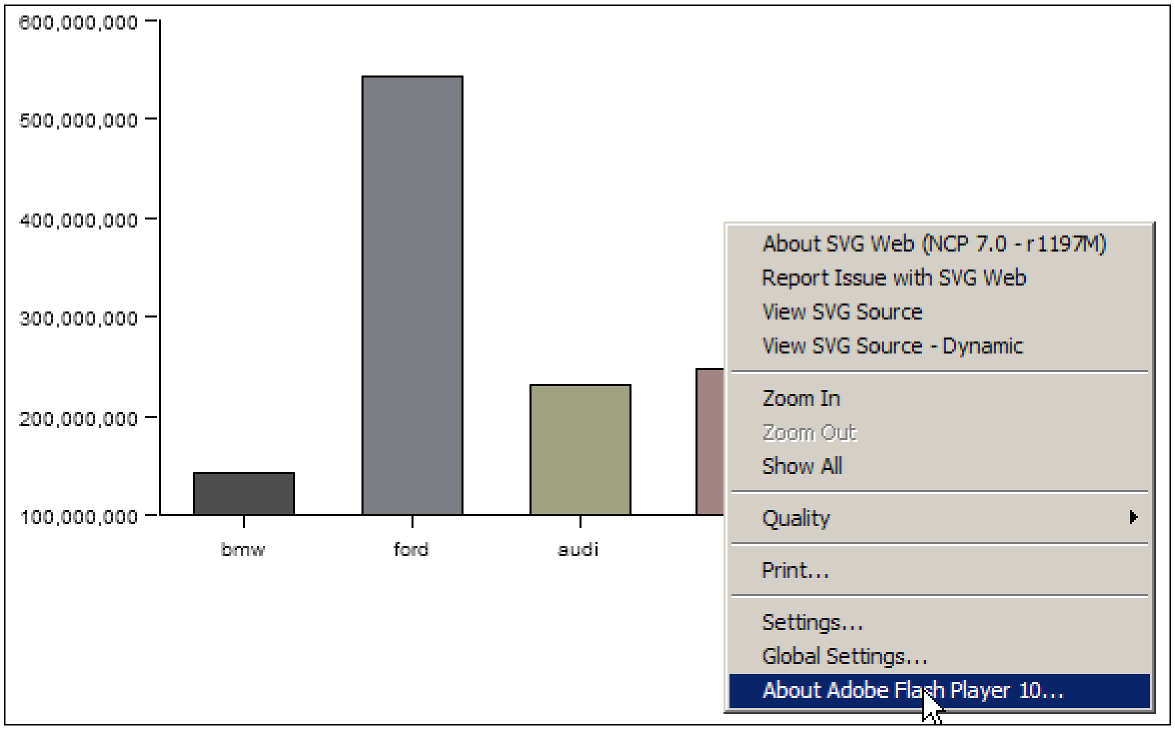


Figure 7 - An SVG chart translated into Flash

9.0 Understanding DataModels

In addition to creating object representatons of the various components of a chart, NetCharts Pro also introduces the concept of DataModels. DataModels are object-based representations of the data to be displayed in the chart. They can be used to provide a standard interface into any data source that can be programmatically accessed. The list of data sources available include:

- any file-based data (XML, CSV, text, etc.)
- any source available via JDBC (most major databases)
- any source available via ODBC (accessed via some JDBC-ODBC bridge or JDBC-ODBC driver)
- any source available via HTTP
- and more…

Various components of NetCharts Pro are used to specifically define the data that will be charted. Some examples would be the `NFBarSet`, `NFStockSet`, `NFLineSet`, etc. Each of these components can be initialized using DataModels using the `loadDataModel(DataModel model)` method of the component. By creating or using an existing DataModel implementation that defines access to the data, a developer can quickly integrate data from other systems into a NetCharts Pro chart.

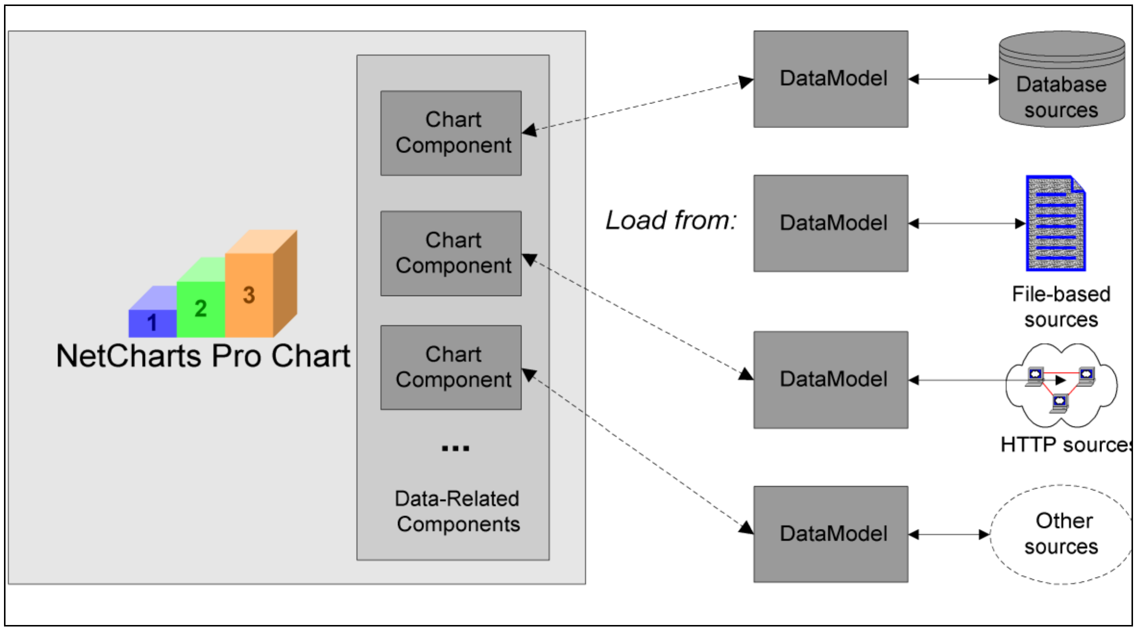


Figure 8 – NetCharts Pro Data Models

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including many with sample DataModel implementations that can be used for access to existing data or as a basis for your own DataModel implementation. To create new DataModels, a developer will need to implement one of the abstract classes in the `netcharts.pro.datamodel` package. Within this package are a set of classes designed for the various scenarios in which data can be represented to the components of NetCharts Pro. Each class is designed for a particular data type (One-Dimensional, Two-Dimensional, Time, Stock, etc.) and includes descriptive information in the JavaDoc API documentation to help developers select the correct class to extend and implement.

Developing a DataModel

The online NetCharts Pro demonstration application at http://ncpro.visualmining.com/ncprowebexamples/ contains multiple sample applications, including an application called *Loading data into a project schedule*, which demonstrates using NetCharts Pro to present an project schedule chart from a CSV file produced by Microsoft Project. The full source code of the example is available from the demonstration application.

This example uses a custom DataModel to load file-based data into various NetCharts Pro components to dynamically generate the chart. The example servlet creates a NetCharts Pro `NFTimechart` instance based upon a default template. Then the servlet uses the `ncpro.examples.datamodel.StringTimeTaskDataModel` to load the CSV based data from the data file into the chart. The chart instance is then used to invoke the NetCharts Pro `getPage()` method to generate the interactive chart image which is returned to the client.

1. Deciding which abstract DataModel class to implement

The first step in creating a custom DataModel is deciding which abstract DataModel class from the set available (in the `netcharts.pro.datamodel` package) to implement. The name of each class describes the data type the class is intended to represent. For instance, the `NFDataModel1D` class should be extended and implemented when the data is One-Dimensional like bar sets (e.g. value, value, value, …). And the `NFDataModelStock` should be used when representing stock values (i.e. open, low, high, close). Each class includes a description in the JavaDoc API to help developers determine the correct class to implement.

The data being accessed by the `ncpro.examples.datamodel.StringTimeTaskDataModel` class will be used to create chart components such as tasks, task labels, etc. This data is related to time and task data, so the `ncpro.examples.datamodel.StringTimeTaskDataModel` class should extend the `netcharts.pro.datamodel.NFDataModelTime` class.

```
      public class StringTimeTaskDataModel extends NFDataModelTime {
      …
```

2. Implementing any abstract methods

Next, the abstract methods of the DataModel class being extended, `netcharts.pro.datamodel.NFDataModelTime` in this case, must be implemented. They are used to load the data being accessed by the custom DataModel into the chart components. The methods of the `netcharts.pro.datamodel.NFDataModelTime` class that need to be implemented are:

- `public Object getStartDateTime(int row)` - Returns the task start date/time related to this row.
- `public Object getStopDateTime(int row)` - Returns the task stop date/time related to this row.
- `public Object getTaskText(int row)` - Returns the text to display related to this task.
- `public Object getTaskColor(int row)` - Returns the color to display on the task element. If null, the default color table will be used to determine the color.
- `public Object getTaskTextStyle(int row)` - Returns the text style to apply to the task element. Allows for custom text styles based on the level of the task.
- `public Object getTaskName(int row)` - Returns the name of the task.
- `public int getNumTasks()` - Returns the total number of points available in the data object.

3. **Adding additional implementation to load and access the data.**

The `StringTimeTaskDataModel` class receives the CSV string data via constructor or the `processString` API method. Custom implementations can handle accessing the source data as appropriate. How a DataModel encapsulates retrieving, parsing and processing data is strictly related to the class implementation alone and do not reflect any needed implementation imposed by the NFDataModel classes. Each custom DataModel implementation may be different based upon the type of data being represented. Two-Dimensional data may use a Vector of Vectors to store the data, or a two-dimensional array. Developers may create additional classes to simplify access to specialized data like stock values.

In the implementation of the `StringTimeTaskDataModel` class, string arrays are used to store the data after parsing the CSV. These string arrays are populated from data that falls within and rows of data. The following code is from the processString method and loads the string arrays from the data:

```
try {
        StringTokenizer st = new StringTokenizer(dataObject, lineDelim);
        if (st == null || st.countTokens() < 1)
                return;

        tasks = new Object[st.countTokens()];
        int cnt=0;
        while (st.hasMoreTokens()){
                String line = st.nextToken();
                StringTokenizer lt = new StringTokenizer(line, fieldDelim);
                String fields[] = null;
                if (lt.countTokens() > 0){
                        fields = new String[lt.countTokens()];
                        int fcnt = 0;
                        while (lt.hasMoreTokens())
                                fields[fcnt++] = lt.nextToken();
                }
                        tasks[cnt++] = fields;
        }
} catch (Exception ex){
        System.out.println("processString failed for "+dataObject+" "+ex);
}
return;
```

After the `StringTimeTaskDataModel` instance is initialized from the data from the CSV file given via the string, the `NFDataModelTime` methods are now able to return the proper data.

10.0 Using NetCharts Pro in Integrated Development Environments

NetCharts Pro can be used in Java Integrated Development Environments such as Eclipse, Sun NetBeans, Oracle JBuilder or IBM WebSphere studio to build, compile and test chart-enabled Java programs.

For example, the following steps can be used to configure an Eclipse project to load, compile and run the NetCharts Pro sample application SimpleExample.java in the examples folder of the NetCharts Pro source distribution.

1) Create a new Java project.
2) Add the NetCharts Pro jar files to the project.
    a. Select the "Libraries" tab in the Java Build Path of the project properties. This can be found in the second page of the "New Java Project Wizard" or via "Project->Properties".
    b. Click "Add External JARs…"
    c. Browse to and select all jars in $NCPROINSTALL/lib.
3) Click the "Finish" button to finish project creation or "OK" to set the project properties.
4) Add the sample test program to the project
    a. Right click on project and add a new package (New->Package).
    b. Name the package *ncpro.examples.standalone*.
    c. Select the package *ncpro.examples.standalone.*
    d. Right click on the package *ncpro.examples.standalone* and select "Import…".
    e. Select "File system" and press "Next >".
    f. Browse to and select the *$NCPROINSTALL/examples/programs/ ncpro/examples/standalone/* directory.
    g. Check *SimpleExample.java*.
    h. Press "Finish".
5) Add the NetCharts Pro license (**NetChartsPro.license**) to the project
    a. Right click on the project folder within the project and select "Import…".
    b. Select "File system" and press "Next >".
    c. Browse to and select the *$NCPROINSTALL/classes/* directory.
    d. Check **NetchartsPro.license**.
    e. Press the Into folder Browse button and select the classes directory.
    f. Press "Finish".
6) Run the sample program
    a. Select "Run->Run…".
    b. Select Java Application
    c. Press "New" to create a new configuration.
    d. Enter an understandable name for the configuration.
    e. Select the correct project.
    f. Press "Search…" to select the main class. Select *SimpleExample*.
    g. Press "Apply".
    h. Press "Run".

The results should look similar to the following.

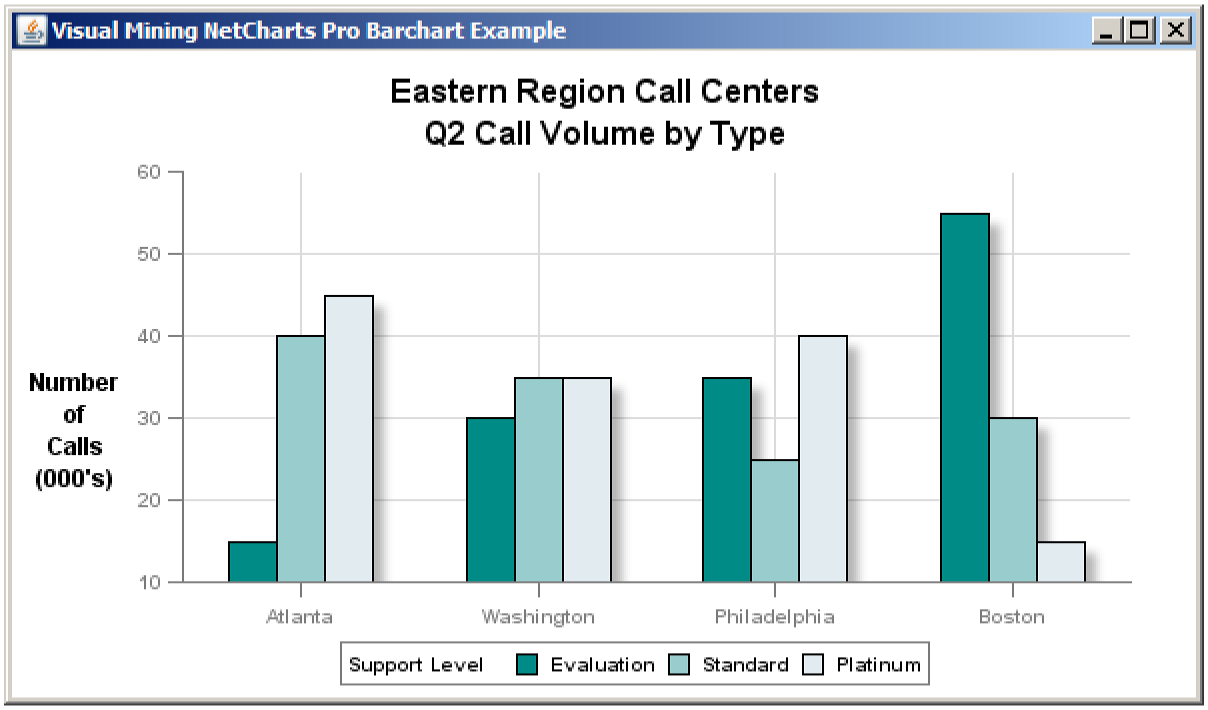


Figure 9 – SimpleExample as shown running from an IDE

Appendix A – Frequently Asked Questions

Q: I'm using NetCharts Pro and in my active labels I try to call a JavaScript function but the URL comes out like: http://javascript/javascript.html?javascript:helloFunction();

A: NetCharts Pro uses java.net.URL instances to create the URL components of ActiveLabels. To properly create a URL with the *javascript* protocol, the JVM that is running within the Web Application must have a protocol handler associated with the *javascript* protocol. If your Application Server does not, then URL's with the *javascript* protocol will fail to be created.

The following lines of code can help determine if a protocol handler is registered for the *javascript* protocol:

```
try {
    URL newURL = new URL("javascript://");
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

If you see:

```
java.net.MalformedURLException: unknown protocol: javascript
    at java.net.URL.<init>(URL.java:497)
    at java.net.URL.<init>(URL.java:390)
    at java.net.URL.<init>(URL.java:344)
    ...
```

then there is no registered protocol.

What you can do is use a Servlet to load a javascript protocol handler within your Web Application. By loading the class within the *init* method of a Servlet that is loaded on startup, the code will be run once before any chart processing occurs. NetCharts Pro includes a *javascript* protocol handler, `netcharts.pro.handler.javascript.Handler`, in the **ncp-js-handler.jar** file included with the NetCharts Pro distribution. In addition, a Java Servlet is included in the **ncp-js-servlet.jar** that can be used to load the protocol handler. Include a standard servlet mapping within your Web Application to load the servlet on startup.

We recommend adding the **ncp-js-handler.jar** file to the highest location (in terms of relative classpath) in the Application Server hierarchy.

General Information

Visual Mining, a division of Tervela, Inc.
2301 Research Blvd.
Suite 201
Rockville, MD 20850

Inquiries

General Phone	800.308.0731
International	+1.301.795.2200
Fax	301.947.8293
General Inquiries	info@visualmining.com
Customer Support	support@visualmining.com
Sales	sales@visualmining.com
Press and Media Inquiries	marketing@visualmining.com

Web
http://www.visualmining.com