NetChartControl v1.0

Reference Manual

Serbia and Montenegro
18000 Nis

phones:
+381 18 521 918
+381 18 322 578
+381 64 21 19 861

online information:
www.netchartcontrol.com
info@netchartcontrol.com

Contents:

What is a NetChartControl?

The **NetChartControl** is a *.NET component* (or a *compiled custom control*) which is used to generate different kinds of charts in .NET applications. It is used for data visualization for the visitors of, i.e. some web page that has to process a statistical data. Below you can see an image example of the generated chart using **NetChartControl**.

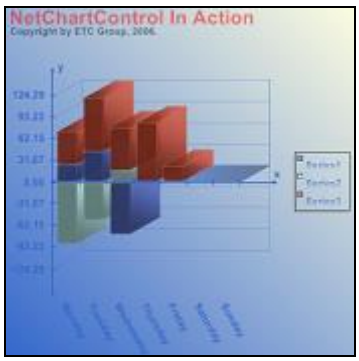

Figure 1: an example chart

NetChartControl is a composite object which contains two more objects: a **NetChartSeries** object which represents a data sample through some referent value period (such is a time period), and a **NetChartAxis** object which shows how data values are changing.

This charting control is developed to be an easy to use and implement in existing web-server pages in .NET applications. It supports all basic chart types (bars, cylinders, lines, areas and pies) and some additional types (doughnuts, pyramids, cones and bubbles). That makes a total of 9 different chart types.

Also, **NetChartControl** is able to generate 2D and 3D charts. It has fully scalable text and graphics rendering engine which can follow the change in the size of the control.

NetChartControl can generate multi-series charts and also a multi-charts (more charts of different types in the same time).

It is possible to show or hide some elements of the chart like: title and subtitle, legend, axis labeling. It also possible to show or hide a certain **NetChartControl** series object.

There are many features that can be changed at any moment of the life cycle of this control. They are described in the sections of this manual that correspond to certain object of the chart.

NetChartControl can be used on the WinForms or in the ASP.NET applications. So, its the unique control for all .NET form-based applications.

It also has built-in XML support to export data and settings to the XML file, or to load it from an XML file.

This control is written totaly in C# language, but it can be used from any .NET language which supports Common Language Runtime Compilation (CLR). It is primarly based on the new Windows graphics library called GDI+ which is a part of the .NET Framework distribution.

This control also uses a double-buffering technique to avoid flickering on the screen during drawing itself.

Component internal structure

The **NetChartControl** is composed of the following:

- Series
- Axis
- Legend
- Title and subtitle
- Background

You can access each of these objects by using provided interface functions[1]. Then you will be able to set a different settings for these objects.

As an example, let's say you have obtained the pointer to the series object. You can change the type of the series (from 9 available chart types mentioned above), or the color of the series, or the name that is show in the legend, etc.

Chart types

There are total of 9 basic chart types that can be generated using **NetChartControl**. This number may sound small but you'll have to add to that number other features like: grouping, exploding, 2D and 3D rendering mode, shading mode, color switching, multicharts, etc. In this way the number of different charts that can be generated can go over 50.

Basic chart types are of the following:

- Bars
- Cylinders
- Lines
- Areas
- Pies
- Doughnuts
- Pyramids
- Cones
- Bubbles

Depending on your needs you will select the type of the chart (or charts) that you want to generate. In this release of the **NetChartControl** pie charts (like doughnuts, pyramids and cones) can show just one value for certain series.

[1] In this release of the **NetChartControl** only series can be retrieved as objects, and you can set params of the other objects using interface of the **NetChart** class (and that is the control itself).

Main class: NetChart

In this section the most important class of the **NetChartControl** component will be described. Here are the lists of the public properties and methods that you can access (with explanation for each property and method).

Attributes list:

- **ShowTitle** (shows or hides title and the subtitle on the generated chart)
- **TitleText** (gets or sets title text)
- **TitleColor** (gets or sets title color)
- **SubtitleText** (gets or sets subtitle text)
- **SubtitleColor** (gets or sets subtitle color)
- **ShowLegend** (shows or hides legend on the generated chart)
- **BackgroundMode** (gets or sets the background mode – gradient, image or default system)
- **FirstBgColor** (gets or sets the first background gradient color)
- **SecondBgColor** (gets or sets the second background gradient color)
- **GradientAngle** (gets or sets the background gradient angle)
- **BackgroundBitmap** (gets or sets the name of the background bitmap)
- **ShowLabels** (shows or hides axis labeling on the generated chart)
- **AxisType** (gets or sets the axis labeling – built in or custom)
- **AxisColor** (gets or sets the color of the axis)
- **SeriesJoin** (gets or sets series grouping mode – only for bars and cylinders)
- **SeriesExplode** (gets or sets series explode mode – only for pies, doughnuts, pyramids and cones)
- **GraphicsMode** (gets or sets charts graphics rendering mode – 2D or 3D)
- **ShadingMode** (gest or sets chart graphics shading mode)
- **ColorMode** (gets or sets chart graphics color mode – colors or black and white)
- **TransparentColor** (gets or sets chart transparent color – useful for web graphics)

Methods list:

- **AddSeries(string seriesName)** – adds a new series to the chart
- **AddSeries(NetChartSeries newSeries)** – adds a new series to the chart
- **GetSeries(int index)** – gets series from the chart
- **GetSeriesNumber()** – gets a number of series from the chart
- **ClearSeries()** – removes all series from the chart
- **Save(string fileName, ImageFormat imageFormat)** – saves a chart image
- **OutputAsBmp()** – saves an image of the generated chart as a .BMP file to the *HttpResponse.OutputStream* object (for use on ASP.NET pages)
- **OutputAsPng()** – saves an image of the generated chart as a .PNG file to the *HttpResponse.OutputStream* object (for use on ASP.NET pages)
- **OutputAsJpeg()** – saves an image of the generated chart as a .JPEG file to the *HttpResponse.OutputStream* object (for use on ASP.NET pages)
- **OutputAsGif()** – saves an image of the generated chart as a .GIF file to the *HttpResponse.OutputStream* object (for use on ASP.NET pages)
- **Print(Rectangle customRectangle)** – shows chart in the print-preview window
- **UpdateChart()** – re-draws the chart on the screen
- **SetAxisLabels(ArrayList xLabelsList)** – sets custom axis labeling
- **LoadXML(string filename)** – loads data and settings from XML file
- **SaveXML(string filename)** – saves data and settings to XML file

You can get or set these properties, or call these methods, anytime you like during the life cycle of the **NetChartControl**. After setting some property, or adding a new series to the chart, you should call **UpdateChart()** method to re-draw the chart control on the screen[2].

[2] This is important only in WinForms apllication.

Main building block: NetChartSeries

Each chart consists of series that hold different values. You create all your series objects using provided **NetChartSeries** class which is a part of the **NetChartControl** namespace. After series is created you can set its params and its data. Here is the list of public properties and methods for this class:

Attributes list:

- **SeriesName** (gets or sets the name of the series that is shown in the legend)
- **Values** (an ArrayList object for holding series data)
- **SeriesType** (gets or sets the type of the series)
- **SeriesColor** (gets or sets the color of the series)
- **Visible** (gets or sets series visibility)

Methods list:

- **Draw(/*...params of this method...*/)** – the one and only public method in the class, but not intended to be used by you.

Now you know the internal structure of this very important class in the **NetChartControl** namespace. It is very easy to manipulate with series data (its well-known ArrayList class). You can add any type of the numerical data you like (integers, floats or doubles). Internaly, all these types are converted to double when drawn on the screen so there will be no data loss.

Working with axis

To set params of the **NetChartControl** axis object you can use interface of the main **NetChart** object (see the list above). You can hide or show axis labeling, change its color or change axis labeling. Here is a list of the built-in labeling:

Labeling list:

- **AxisTypeMonths** (the months list – from January to December)
- **AxisTypeWeekdays** (the weekday list – from Monday to Sunday)
- **AxisTypeWorkdays** (the workday list – from Monday to Friday)
- **AxisType12Hours** (the hours list – from 01:00 to 12:00)
- **AxisType24Hours** (the hours list – from 01:00 to 24:00)
- **AxisTypeCustom** (the custom list – anything you want)

Working with chart title and subtitle

It is very simple using **NetChart** clas interface. You can set chart title and subtitle text, its color and to show or hide it on the chart. This is all stated in the above list of public properties of the **NetChart** class.

Working with chart legend

Again, using simple interface of the **NetChart** class you can show or hide the chart legend.

Working with background modes

You can have a gradient background at any angle you like (or no gradients if you use the same color value for both background gradient colors), an image on the chart background or a default system background. This really affects the visual look of the chart (together with the 3D series look). Below is the list.

Background mode list:

- **BackgroundModeGradient** (background is a gradient color)
- **BackgroundModeImage** (background is an image)
- **BackgroundModeDefault** (background is a default system control background)

Working with color modes

The **NetChartControl** can have its output in several different color modes which apply some basic color processing features that don't hit peformance to much. These modes are shown below.

Color mode list:

- **ColorModeRGB** (output is in a normal colored mode)
- **ColorModeBW** (output is in a gray scale – better for printing on non-color printers)
- **ColorModeSepia** (output is in a old-movie style)

Exporting chart image

You can save the image of the generated chart in many different formats (actualy the one that **Save()** method of the **Bitmap** system class supports). The list shows all of them.

Image export file formats:

- **BMP** (saved image is in a bitmap format)
- **EMF** (saved image is in a enhanced Windows metafile format)
- **EXIF** (saved image is in a exchangeable image format)
- **GIF** (saved image is in a graphics interchange format)
- **ICO** (saved image is in a ican format)
- **JPEG** (saved image is in a joint photographic expert image format)
- **PNG** (saved image is in a portable network graphics format)
- **TIFF** (saved image is in a tag image format)
- **WMF** (saved image is in a Windows metafile format)

The most common formats used on the Internet today are JPEG and GIF file formats, so you will probably save your charts in ASP.NET pages in one of these formats.

Printing charts

You can also print generated charts. Use **Print()** method of the **NetChart** class to open print-preview window from where you can print chart image.

XML support

The **NetChartControl** can import/export data and settings from/to an XML file, using provided interface functions **LoadXML()** and **SaveXML()**. In this way a small files are generated and can be transfered across the Internet to be used on some other place, instead of transfering large images.

WinForms example application

Here is an example WinForm application that uses **NetChartControl** to generate some charts.

At the begining of each form *.cs* file you must state the following include statement to show compiler that you will use classes from **NetChartControl** namespace.

```
using NetChartControl;
```

Next, you should add reference to your project file to the folder the *NetChartControl.dll* is. It is well-known procedure and I will just state basic steps here:

- right on the project name in your Solution Explorer
- select Add Reference... button
- use Browse button and find compiled **NetChartControl** module
- select the module in the list
- select OK button and the reference will be inserted in your project

You should be able, in the Design mode, to drag-and-drop **NetChartControl** on the form directly, so that will generate the contruction code in the form. You should see the line at the beginning of the form file similar to this one:

```
private NetChartControl.NetChart netChart1;
```

If everything went well in the previous steps you have finished the hardest part. Now, all you have to do is create your series somewhere in the form class. I used **Load()** method of the form in the sample project, but you can do what ever you like. Here is how you can create your series:

```
NetChartSeries series1 = new NetChartSeries();
series1.SeriesName = “Series 1”;
series1.SeriesType = NetChartSeriesType.SeriesTypeCylinder;
series1.SeriesColor = Color.Blue;
series1.Values.Add(100.0);
series1.Values.Add(-10.0f);
series1.Values.Add(5);
```

That’s it. This series is created and ready to be added to the chart. See how below:

```
netChart1.AddSeries(series1);
```

You can repeat this procedure and create as many series as you like (but not too much because you should see something on the screen). Then you can set the params of the chart itself like this:

```
netChart1.AxisType = NetChartAxisType.AxisTypeWeekdays;
netChart1.SeriesJoin = NetChartSeriesJoin.SeriesJoinGroup;
netChart1.SeriesExplode = NetChartSeriesExplode.SeriesExplodeFar;
netChart1.GraphicsMode = NetChartGraphicsMode.GraphicsMode3D;
netChart1.ShadingMode = NetChartShadingMode.ShadingMode1;
netChart1.ShowLabels = true;
netChart1.ShowLegend = true;
netChart1.ShowTitle = true;
netChart1.UpdateChart();
```

And this should work with no problems. But you must experint to get the results you want, after all.

ASP.NET example application

Take a look now at the ASP.NET example application which uses this control to generate dynamic charts on the web page. The first part (adding a reference to the project and putting the control on the WebForm) is the same as in WinForm example application above, so I wont repeat it again.

Suppose you have added correct reference and dragged control to the WebForm you will have the same series initialization stuff as above, but this time I selected **Page_Load** event. I'll just mention here that you should put code that defines the size of the **NetChartControl** since it will be the size of the generated bitmap. Like this:

```
netChart1.Width = 500;
netChart1.Height = 500;
```

After you add your series there is no other way to show the chart but to render it as an image file. Here is the code at the **Page_Load** event that generates a JPEG image:

```
netChart1.OutputAsJpeg(Response);
```

Or if you more prefer GIF images:

```
netChart1.OutputAsGif(Response);
```

You should put this image creation code on the separate page, let's say *ImageGenerator.aspx*. The HTML calling code on the other page, ie. *Results.aspx* (where you want your image to be shown) would be:

```
<img src="ImageGenerator.aspx">
```

You should have a working page now which shows a chart image to the visitor.

Summary

In this short reference manual, all features of the current implementation of the **NetChartControl** are shown. For all your questions or request please use the provided contact information on the first page of this manual.

This release of the **NetChartContol** has passed a long period of testing so there are no reported or known bugs for now. If this is your case, please send all information considering found bugs to the info@netchartcontrol.com in order make a review and to have it repaired as soon as possible.

Regards,

ETC Group Team.
2006.