

THE JAVA DEVELOPERS' GUIDE TO BUSINESS REFINERY SimpleCharts SDK

Version 3.0

Last updated on April 25, 2007

ALL RIGHTS RESERVED BY Business Refinery © 2001, 2007.

Table of Contents

1 Introduction.....	4
1.1 About SimpleCharts SDK.....	4
1.2 Features of SimpleCharts SDK	4
1.3 Components of SimpleCharts SDK	5
1.4 SimpleCharts SDK for Java Installation	5
1.5 Development Environment Setup	6
1.6 Run Demo Application	6
1.7 Compatibility	6
2. SimpleCharts SDK in Java	7
2.1 Jump Start	7
2.2 Common Procedures to Create a Chart	8
2.3 Bullet Graph.....	8
2.3.1 The Dataset.....	8
2.3.2 Chart Settings	9
2.3.3 Constructing the Chart	9
2.3.4 Save into Image File	9
2.3.5 Customizing Bullet Graph	10
2.4 Sparklines	10
2.4.1 The Dataset.....	10
2.4.2 Chart Settings	11
2.4.3 Constructing the Chart	11
2.4.4 Save into Image File	12
2.4.5 Customizing Sparklines.....	12
2.5 Single Stacked Bar	13
2.5.1 The Dataset.....	13

2.5.2 Chart Settings	13
2.5.3 Item Label Positions	14
2.5.3 Constructing the Chart	15
2.5.4 Save into Image File	15
2.5.5 Customizing Single Stacked Bar	16
2.6 Strip Plot.....	16
2.6.1 The Dataset.....	17
2.6.2 Chart Settings	17
2.6.3 Constructing the Chart	17
2.6.4 Save into Image File	18
2.6.5 Customizing Strip Plot	18
3. Support and Professional Services	19
3.1 Support Web Site	20
3.2 Basic Support.....	20
3.3 Premium Support Services	20
3.4 Professional Services	20

1 Introduction

1.1 About SimpleCharts SDK

Embedded a high performance charting engine, SimpleCharts SDK is Java charting development kit that can be used on Java application and Java web application.

1.2 Features of SimpleCharts SDK

SimpleCharts can generate Bullet Graph, Sparklines, Strip Plot, and Single Stacked Bar.

Additional features include:

- Data is accessible from any implementation of the defined interfaces;
- Works in applications, servlets, JSP (with build in tags) and applets;
- Export to PNG and JPEG;
- Easily design chart colors through color scheme interfaces implementation;
- **Easy of Use**
We strive to make the developer's life easier. Complex parameter configurations are removed from SimpleCharts SDK. You only have to supply most basic settings like data, chart height and width. SimpleCharts can intelligently determine the best setting internally.

SimpleCharts is written entirely in Java, and should run on any implementation of the Java 2 platform (JDK 1.5 or later).

1.3 Components of SimpleCharts SDK

SimpleCharts SDK comprises two essential components:

A core library for Java application development

- SimpleCharts-3.0-core.jar. This jar contains all necessary to create charts with SimpleCharts SDK.

One Java package for Java application and web application development

- SimpleCharts-3.0-all.jar. This package contains all files in SimpleCharts-3.0-core.jar and some JSP tag library for developer to create chart in JSP/Servlet environment.

For more information, please go to <http://www.businessrefinery.com/products/downloads.html>

1.4 SimpleCharts SDK for Java

Installation

First, make sure that you have already installed JDK 1.5/5.0 or above on your system. If you are running under server environment, you need also install Tomcat 5.5 or above to run demo application. SimpleCharts can be run on any J2EE environment, and our demo application is based on Tomcat 5.5)

Download a copy of SimpleCharts SDK from <http://www.businessrefinery.com/products/downloads.html>. Simply unzip it to an empty folder. Let refer this folder as CHART_HOME.

The file organization of SimpleCharts SDK distribution is as follows:

CHART_HOME

- javadoc [Java docs]
- SimpleCharts-3.0-core.jar [jar for java application]

- SimpleCharts-3.0-all.jar [jar for java application and web application]
- DevelopersGuide.pdf [The developer's guide]
- demo-src [all demo java source codes]
- demo-web [one sample demo jsp application and their source code]
- LICENSE-EVALUATION-SIMPLECHARTS.txt [License agreement]
- Purchase-SimpleCharts.html [Click to purchase SimpleCharts]

1.5 Development Environment Setup

After you have obtained and unzipped the SimpleCharts SDK kit, you need to setup your development environment in order to develop Java applications with SimpleCharts SDK. To do so, you need:

- Set SimpleCharts-3.0-core.jar into your class path (If you only need core library to create charts and do not require jsp tag library to create chart in JSP)
- Put SimpleCharts-3.0-all.jar into your class path (This jar need Java web development environment, make sure jsp-api.jar and servlet-api.jar are in your class path also. Those two jars can be found in \${TOMCAT_HOME}/common/lib.

1.6 Run Demo Application

To run demo codes for each types of chart, you need install JDK 1.5 or above.

To run web application, you need install TOMCAT 5.5 or above. Please copy folder CHART_HOME/demo-web/simplecharts to TOMCAT_HOME/webapps/.

1.7 Compatibility

Operating Systems: Windows, Linux, Max OS, any platforms support Java
Java Runtime: Version 1.5 (5.0) or above.

2. SimpleCharts SDK in Java

2.1 Jump Start

The following code demonstrates the basic usage of SimpleCharts to create a bullet graph.

```
1  package demo.bulletgraph;
2
3  import com.businessrefinery.simplecharts.bulletgraph.ChartOption;
4  import com.businessrefinery.simplecharts.bulletgraph.BulletGraphChart;
5  import com.businessrefinery.simplecharts.bulletgraph.DefaultBulletGraphDataset;
6
7  public class BulletGraphDemo
8  {
9      public static void main(String[] args)
10     {
11         DefaultBulletGraphDataset dataset = new DefaultBulletGraphDataset(true);
12         String item = "USA";
13         dataset.addMeasureValue(50, item);
14         dataset.addTargetValue(53, item);
15         dataset.addRange1Value(25, item);
16         dataset.addRange2Value(40, item);
17         dataset.addRange3Value(60, item);
18
19         ChartOption chartOption = new ChartOption(
20             220,                // chartWidth
21             40,                 // chartHeight
22             dataset);           // dataset
23
24         BulletGraphChart bulletGraph = new BulletGraphChart(chartOption);
25
26         bulletGraph.createChart("C://bullet_graph.png");
27     }
28 }
29
```

Line 3-5: Imports all classes required;

Line 11: Create a DefaultBulletGraphDataset object, which will contains all data displayed in the chart.

Line 13-17: Add data to dataset

Line 19-22: Create a ChartOption object, which allows developer to customize chart display. Here we only pass three required attributes: chart width (in pixel), chart height (in pixel), and dataset.

Line 24: Create BulletGraphChart object with the chartOption.

Line 26: Save bullet graph object into bullet_graph.png under C drive.

For a complete sample application, please refer to CHART_HOME/demo-src.

2.2 Common Procedures to Create a Chart

1. Create a dataset for chart.
2. Create a ChartOption object for chart, to define all your customization of that chart
3. Create a Chart object with that ChartOption object
4. Call createChart(filename) method in Chart object, to save chart into PNG/JPEG format.

2.3 Bullet Graph

2.3.1 The Dataset

The first step in generating the chart is to create a dataset. For bullet graph chart, you need create a DefaultBulletGraphDataset object, which is implementing IDataset interface.

```
public IDataset createDatasets() {  
    DefaultBulletGraphDataset dataset = new  
        DefaultBulletGraphDataset(true);  
  
    dataset.addMeasureValue(50000, "USA");  
    dataset.addTargetValue(53000, "USA");  
    dataset.addRange1Value(25000, "USA");  
    dataset.addRange2Value(40000, "USA");  
    dataset.addRange3Value(60000, "USA");  
    return dataset;  
}
```

Notice that we have used String objects as the item keys for the data values.

2.3.2 Chart Settings

ChartOption class contains all settings for bullet graph. There are three required settings for bullet graph chartWidth (int), chartHeight (int), and dataset (IDataset). The others are all optional and there are default values for each of them. For more information, please go to java docs.

```
private ChartOption getBasicChartOption() {  
    return new ChartOption(  
        220,          // chartWidth in pixel  
        320,          // chartHeight in pixel  
        createDatasets()); // dataset  
}
```

2.3.3 Constructing the Chart

```
public JFreeChart getChart() {  
    BulletGraphChart bulletGraph = new BulletGraphChart(chartOption);  
    return bulletGraph.createChart();  
}
```

It constructs a JFreeChart with chart options. The **chartOption** is the one created in the previous section.

2.3.4 Save into Image File

There are two ways to save chart file. One is

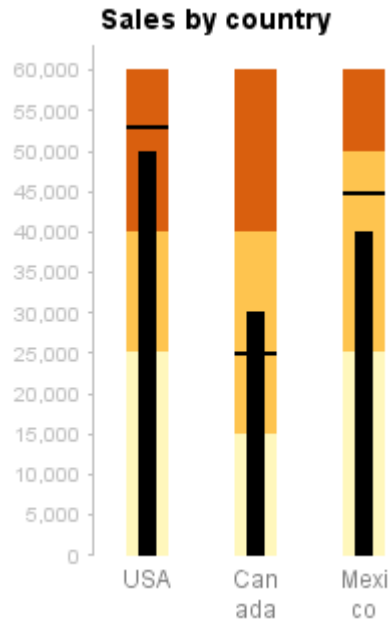
```
bulletGraph.createChart("C:\\BulletGraph.png");
```

or

```
bulletGraph.createChart("C:\\BulletGraph.jpg");
```

Now a new bullet graph chart named BulletGraph.png or BulletGraph.jpg will be created under your C drive. See. That's simple!

Here is the sample chart



2.3.5 Customizing Bullet Graph

You can customize bullet graph chart through ChartOption object's attributes. To know more about them, please go to [java doc](#) to get more information

2.4 Sparklines

2.4.1 The Dataset

The first step in generating the chart is to create a dataset. For sparklines chart, you need create a DefaultSparklinesDataset object.

```
protected DefaultSparklinesDataset createChartDataset() {  
    DefaultSparklinesDataset dataset = new DefaultSparklinesDataset();  
    dataset.setItemValue(212D, "Jan");  
    dataset.setItemValue(504D, "Feb");  
    ...  
    return dataset;  
}
```

Notice that we have used String objects as the item keys for the data values.

2.4.2 Chart Settings

ChartOption class contains all settings for sparklines. There are three required settings for sparklines chartWidth (int), chartHeight (int), and dataset (IDataset). The others are all optional and there are default values for each of them. For more information, please go to java docs.

```
private ChartOption getBasicChartOption() {  
    return new ChartOption(  
        150, // chart width  
        35, // chart height  
        createChartDataset()); // dataset  
}
```

2.4.3 Constructing the Chart

```
public JFreeChart getChart() {  
    SparklinesChart sparklines = new SparklinesChart (chartOption);  
    return sparklines.createChart();  
}
```

It constructs a JFreeChart with chart options. The **chartOption** is the one created in the previous section.

2.4.4 Save into Image File

There are two ways to save chart file. One is

```
sparklines.createChart("C:\\Sparklines.png");
```

or

```
sparklines.createChart("C:\\Sparklines.jpg");
```

Now a new bullet graph chart named Sparklines.png or Sparklines.jpg will be created under your C drive. The **sparklines** is the one created in the previous section.

Here is the sample chart



2.4.5 Customizing Sparklines

You can customize sparklines chart through ChartOption object's attributes. To know more about them, please go to java docs to get more information

2.5 Single Stacked Bar

2.5.1 The Dataset

The first step in generating the chart is to create a dataset. For single stacked bar chart, you need create a `DefaultSingleStackedBarDataset` object.

```
protected IDataset createChartDataset()
{
    DefaultSingleStackedBarDataset dataset = new
        DefaultSingleStackedBarDataset();
    dataset.setItemValue(15.76D, "Labour");
    dataset.setItemValue(8.66D, "Administration");
    dataset.setItemValue(4.00D, "Marketing");
    dataset.setItemValue(3.51D, "Distribution");

    return dataset;
}
```

Notice that we have used String objects as the item keys for the data values.

2.5.2 Chart Settings

`ChartOption` class contains all settings for the chart. There are three required settings for it, `chartWidth` (int), `chartHeight` (int), and `dataset` (`IDataset`). The others are all optional and there are default values for each of them. For more information, please go to java docs.

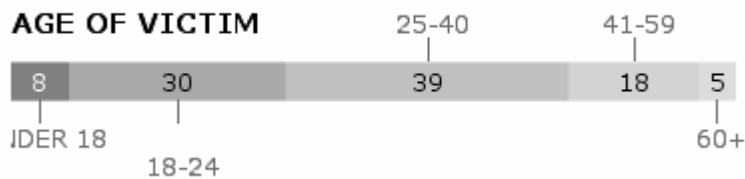
```
private ChartOption getBasicChartOption() {
    return new ChartOption(
        400, // chart width
        175, // chart height
        createChartDataset()); // dataset
}
```

2.5.3 Item Label Positions

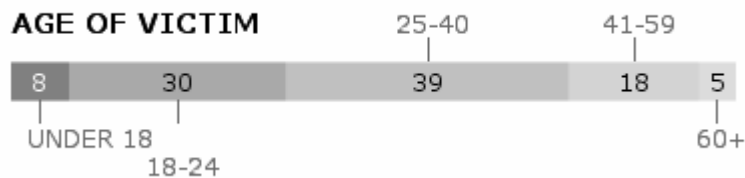
In order to display item's label properly (no overlapping between two labels and avoid chopping by chart borders), developer can adjust label's position through `DefaultItemLabelPositionSetting` object.

There are five attributes for developer to set: `adjustFirstItem` (boolean), `adjustLastItem` (boolean), `aboveItemsTwoLevelIndent` (boolean), `belowItemsTwoLevelIndent` (boolean) and `postionInfos` (int[]).

`adjustFirstItem`: by default, item label's horizontal position base point is in the center of the item bar and label alignment is center of the base point. If bar size is small and label length is too large, some chars in the beginning will be chopped by the bar border, in such case, developer can reset item label's alignment to the left of the base point through setting `adjustFirstItem` to true.



Above chart, `adjustFirstItem` is false, some chars in first item "UNDER 18" are not displayed.



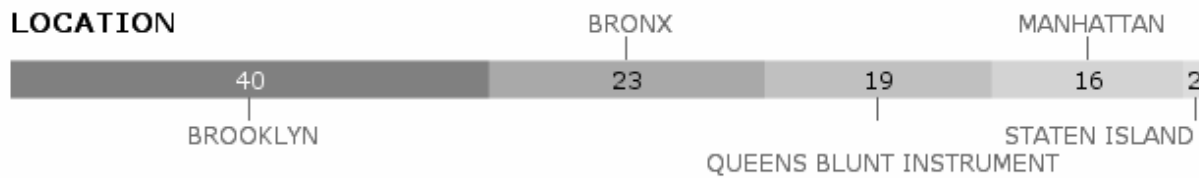
After setting `adjustFirstItem` to true, now item label "UNDER 18" is displayed completely.

`adjustLastItem`: similar with `adjustFirstItem`.

If two item's label's position are too close, they will be overlapped each other, to avoid that, developer can set two levels of indent for item's vertical position.

`aboveItemsTwoLevelIndent`: if true, items above bar will have two levels of intent.

`belowItemsTwoLevelIndent`: if true, items below bar will have two levels of intent.

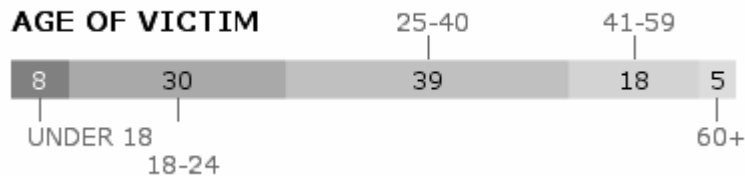


One sample to set belowItemsTwoLevelIndent to true, and set adjustLastItem to true.

postionInfos (int[]):

By default item's labels will be one below the bar, next above the bar, next below the bar, and so on. Developer can manually set each item's position below or above the bar through postionInfos.

Developer can set value an array of integers to postionInfos. In the array, 1 for above, -1 for below, 0 for default. If the length of the array is less than total count of items in the chart, the rest of items' position will be set by default automatically.



Above example, set first two items' positions to be below the bar, to avoid overlapping with chart title.

2.5.3 Constructing the Chart

```
public JFreeChart getChart() {
    SingleStackedBarChart singleStackedBar = new
        SingleStackedBarChart(chartOption);
    return chart.createChart();
}
```

It constructs a JFreeChart with chart options. The **chartOption** is the one created in the previous section.

2.5.4 Save into Image File

There are two ways to save chart file. One is

```
singleStackedBar.createChart("C:\\SingleStackedBar.png");
```

or

```
singleStackedBar.createChart("C:\\SingleStackedBar.jpg");
```

Now a new single stacked bar chart named SingleStackedBar.png or SingleStackedBar.jpg will be created under your C drive. The **singleStackedBar** is the one created in the previous section.

Here is the sample chart



2.5.5 Customizing Single Stacked Bar

You can customize Single Stacked Bar chart through ChartOption object's attributes. To know more about them, please go to java docs to get more information

2.6 Strip Plot

2.6.1 The Dataset

The first step in generating the chart is to create a dataset. For strip plot chart, you need create a `DefaultStripPlotDataset` object.

```
protected IDataset createChartDataset() {
    DefaultStripPlotDataset dataset = new DefaultStripPlotDataset();
    dataset.setItemValue(15.76D, "Labour");
    dataset.setItemValue(4.66D, "Administration");
    dataset.setItemValue(8.00D, "Marketing");
    dataset.setItemValue(6.51D, "Distribution");
    return dataset;
}
```

Notice that we have used String objects as the item keys for the data values.

2.6.2 Chart Settings

`ChartOption` class contains all settings for the chart. There are three required settings for it, `chartWidth` (int), `chartHeight` (int), and `dataset` (`IDataset`). The others are all optional and there are default values for each of them. For more information, please go to java docs.

```
private ChartOption getBasicChartOption() {
    return new ChartOption(
        275, // chart width
        400, // chart height
        createChartDataset()); // dataset
}
```

2.6.3 Constructing the Chart

```
public JFreeChart getChart() {
    ChartOption chartOption = (ChartOption)getChartOption();
    StripPlotChart stripPlot = new StripPlotChart(chartOption);
}
```

```
        return stripPlot.createChart();  
    }
```

It constructs a JFreeChart with chart options. The **chartOption** is the one created in the previous section.

2.6.4 Save into Image File

There are two ways to save chart file. One is

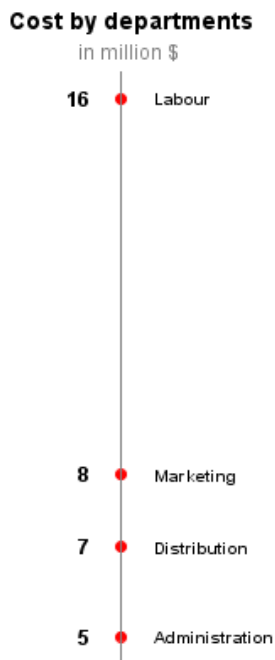
```
stripPlot.createChart("C:\StripPlot.png");
```

or

```
stripPlot.createChart("C:\StripPlot.jpg");
```

Now a new strip plot chart named StripPlot.png or StripPlot.jpg will be created under your C drive. The **stripPlot** is the one created in the previous section.

Here is the sample chart



2.6.5 Customizing Strip Plot

You can customize Strip Plot chart through ChartOption object's attributes. To know more about them, please go to java docs to get more information.

3. Support and Professional Services

3.1 Support Web Site

<http://www.businessrefinery.com/products/simplecharts/support.html>

3.2 Basic Support

Our team provides basic support for general SimpleCharts developers. Email your technical questions to support@businessrefinery.com (Our team may reject your improper enquires without any reply. To avoid this, here is a little piece of advice: You are strongly recommended to subscribe our premium support service in order to get your problems solved quickly.

3.3 Premium Support Services

Free premium support services subscription for a fixed period with every license purchased. You may optionally extend premium support services after your free subscription expires. For more details, check the order page of the product.

3.4 Professional Services

Our team is ready to help you to develop various application and components. Please send your query to info@businessrefinery.com

