**Chapter 3. Basic chart functionality**

In the folder chartdoc of the package you downloaded you can find the documentation, generated by javadoc, which describes packages ua.net.freecode.chart and other packages. For your application-level programs, you need only classes and methods from package ua.net.freecode.chart. We are going to support the compatibility of this package in future versions, this package is the same for both Vaadin 6 and Vaadin 7 and all its public methods and enums are supposed to be used in your code.

All the other packages (ua.net.freecode.chart.impl and others) in this library are not supposed for the direct use, are subject to change in further versions without the support of the compatibility and depend on the Vaadin underlying layer (Vaadin 6 or Vaadin 7).

Package ua.net.freecode.chart has the following classes:

|  |  |
| --- | --- |
| Chart | It is the main class, which you instantiate once for each chart on the web page. Chart is Vaadin Component, so you can add it to any panel, layout or window.  It also contains a lot of methods to customize the chart.  Example:  Chart chart1 = new Chart();  leftPanel.addComponent(chart1); |
| AxisSystem | This library supports many axes (up to 16 axes at the top, up to 16 axes at the bottom, up to 16 axes at the left and up to 16 axes at the right). Of course, you do not need to reach this limit (all 16 axes at any side) otherwise your chart will become cumbersome. When you place your data, the library needs to know what axes your data are related. For this purpose we introduced AxisSystem, which binds your data and your setting to a specific couple of axes. You cannot instantiate it directly, use Chart.createAxisSystem method to do it. |
| ChartPosition | Many items to be placed on the chart require x,y coordinates to place them. We made a flexible system, so that you can use either absolute or relative coordinates, which are relative with respect to top, bottom, left, right, center of the main chart or top, bottom, left, right, center of the chart’s data area and can be expressed either in pixels or in percent to the whole chart length (width or height). ChartPosition encapsulates this functionality. |
| Marker | Determines the shape of the marker to be placed for each x,y points in your data. There are ready markers: circle, square, ring, triangle upward, triangle downward, triangle leftward, triangle rightward, rectangle and even you can define your own markers using SVG path description (this is HTML5 standard) |
| CurvePresentation | This class determines many important aspects related to the presentation of each series. The supported presentations by the present time are a line, single points, area, vertical bars (you can see the ready chart with our line, area and vertical bars in the Introduction). Also you specify the thickness of the line, marker, pies and the Marker mentioned above. You can have curves with different thickness, for example, 1, 2, 3 and so on. |

Now we demonstrate the use of this library to show the chart, which you might view in the Introduction.

Consider the following problem:

We need to present a chart with 4 series, with the X axis data of months from January to December.

The first and second series has the shape of vertical bars, the third series has the shape of line having the thickness of 2, the fourth series is an area having the border thickness of 1.

The code to solve this problem is as follows:

chart = **new** Chart();

//adding the style name below is related to choosing the color scheme and discussed in Chapter 4

chart.addStyleName("UniqueColorsBlueGreenRedScheme");

chart.setWidth("100%");

chart.setHeight("400px");

//20 pixels at the right is necessary because we have long labels at the bottom axis and when

//there is no legend, they are cut if we do not provide some additional space at the right

chart.setMinRightMargin(20);

//our labels at the bottom axis are at the angle of 45 and they need 100-pixel height of the space

chart.setAxisLabelStringHeight(100);

chart.setGeneralTitle("Sample General Chart Header");

chart.setLegendData(**new** String[]{"Freecode","OST Ltd.","IT Ltd.","Dobryvechir"});

//The value of legend item width is adjusted according to the longest string in the legend data,

//which can be chosen experimentally

chart.setLegendItemWidth(112);

AxisSystem axisSystem = chart.createAxisSystem(AxisHorizontal.*Bottom*, AxisVertical.*Left*);

//our range is 0..1000 (it is not necessary to specify 0 as minimum since it is the default)

axisSystem.setVerticalAxisMaxValue(1000);

//the angle for each label at the bottom axis is 45 degrees

axisSystem.setHorizontalAxisLabelAngle(45);

axisSystem.setHorizontalAxisTitle("Year 2012");

axisSystem.setVerticalAxisTitle("Incomes and losses of the company");

axisSystem.setCurvePresentation(**new** CurvePresentation[]{

**new** CurvePresentation(**new** Marker(Marker.MarkerShape.*Circle*),0,CurvePresentation.CurveKind.*VerticalBar*),

**new** CurvePresentation(**new** Marker(Marker.MarkerShape.*Rectangle*),0,CurvePresentation.CurveKind.*VerticalBar*),

**new** CurvePresentation(**new** Marker(Marker.MarkerShape.*Square*),2,CurvePresentation.CurveKind.*Line*),

**new** CurvePresentation(**new** Marker(Marker.MarkerShape.*NoMarker*),1,CurvePresentation.CurveKind.*Area*),

});

axisSystem.setXDiscreteValues(**new** String[]{"January 2012",

"February 2012","March 2012","April 2012","May 2012","June 2012",

"July 2012", "August 2012","September 2012","October 2012",

"November 2012","December 2012"});

axisSystem.setYDiscreteValuesForAllSeries(**new** **double**[][]{

**new** **double**[]{300,400,450,500,657,450,230,100,500,200,300,500},

**new** **double**[]{196,20,212,302,0,12,30,33,64,100,200,212},

**new** **double**[]{0,100,1500,1750,187,155,-190,1900,199,1200,-5,300,-5,300},

**new** **double**[]{0,72,500,100,20,100,500,88,150,160,10,200}

});

That is all as for the chart itself what we need for this problem.

Now at the import part we need to put the code as follows:

**import** ua.net.freecode.chart.AxisSystem.AxisHorizontal;

**import** ua.net.freecode.chart.AxisSystem.AxisVertical;

**import** ua.net.freecode.chart.AxisSystem;

**import** ua.net.freecode.chart.Chart;

**import** ua.net.freecode.chart.CurvePresentation;

**import** ua.net.freecode.chart.Mark;

**import** com.vaadin.Application;

**import** com.vaadin.data.Property;

**import** com.vaadin.data.Property.ValueChangeEvent;

**import** com.vaadin.ui.\*;

In the variables we need to describe the chart variable:

Chart chart

In the main code we need to insert the chart component to our panel, layout or window.

For example, window.addComponent(chart);

That is all.

The full source you can find in folders sample1forvaadin6 for Vaadin 6 and sample1forvaadin7 for Vaadin7.

If you get the source of those samples, you need to make the following steps to run the application:

Instructions for Vaadin6 sample project

1. Create a new Vaadin 6 project in Eclipse

2. Replace your project's src folder with this src folder

3. Replace your project's WebContent/WEB-INF/web.xml file

with this web.xml file

4. Copy

chartlib/charts/d3.v3.min.js to <YOUR PROJECT>/WebContent/VAADIN/charts/d3.v3.min.js

chartlib/charts/chart\_1.0.js to <YOUR PROJECT>/WebContent/VAADIN/charts/chart\_1.0.js

chartlib/charts/fchart.css to <YOUR PROJECT>/WebContent/VAADIN/charts/fchart.css

chartlib/libforvaadin6/fchart-1.0.jar to <YOUR PROJECT>/WebContent/WEB-INF/lib/fchart-1.0.jar

Note: if any folder does not exist in your project, create it!

5. Refresh the project in Eclipse (left click on the project and press F5)

6. Specify that fchart-1.0.jar is one of required libraries:

Right-click on the project in Eclipse -> Properties -> Java Build Path

-> Libraries -> Add JARs -> find fchart-1.0.jar in your project tree

-> Ok

Also in Java Build Path -> Order and Export -> check fchart-1.0.jar

7. Compile your java code (for example, Project -> Clean -> your project,

check immediate rebuild)

Note: if you have the error: internal validation error for d3.v3.min.js, see Trouble-Shooting Guide to solve this problem.

8. Compile Vaadin widgets (Press button in Eclipse: Compile Vaadin Widgets) (this is necessary to do only once and when you develop your application code you do not need to compile Vaadin widgets each time).

9. Run the application on the server (for example, on Tomcat server)

Instructions for Vaadin7 sample project

1. Create a new Vaadin 7 project in Eclipse

2. Replace your project's src folder with this src folder

3. Copy

chartlib/libforvaadin7/fchart-1.0.jar to <YOUR PROJECT>/WebContent/WEB-INF/lib/fchart-1.0.jar

chartlib/libforvaadin7/fchart-1.0.src.jar to <YOUR PROJECT>/WebContent/WEB-INF/lib/fchart-1.0.src.jar

Note: if any folder does not exist in your project, create it!

4. Refresh the project in Eclipse (left click on the project and press F5)

5. Specify that fchart-1.0.jar is one of required libraries:

Right-click on the project in Eclipse -> Properties -> Java Build Path

-> Libraries -> Add JARs -> find fchart-1.0.jar in your project tree

-> Ok

Also in Java Build Path -> Order and Export -> check fchart-1.0.jar

6. Specify that fchart-1.0.src.jar is one of required libraries:

Right-click on the project in Eclipse -> Properties -> Java Build Path

-> Libraries -> Add JARs -> find fchart-1.0.src.jar in your project tree

-> Ok

Leave fchart-1.0.src.jar unchecked in Order and Export

Note: fchart-1.0.src.jar is necessary only for widget compilation

and as a source code for your pleasure.

7. Compile your java code (for example, Project -> Clean -> your project,

check immediate rebuild)

Note: if you have the error: internal validation error for d3.v3.min.js, see Trouble-Shooting Guide to solve this problem..

8. Compile Vaadin widgets (Press button in Eclipse: Compile Vaadin Widgets)

9. Run the application on the server (for example, on Tomcat)

Note: the Eclipse's embedded browser does not show the chart correctly,

use any modern browser.

In the next chapter you can see how to set up colors using css file and dynamically switch color schemes in your code.