



IBM ILOG JViews Charts V8.6

Using the Designer

Copyright

Copyright notice

© Copyright International Business Machines Corporation 1987, 2009.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Trademarks

IBM, the IBM logo, ibm.com, WebSphere, ILOG, the ILOG design, and CPLEX are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Notices

For further copyright information see `<installdir>/license/notices.txt`.

Table of contents

Getting Started.....	5
Development process.....	6
Running the Designer.....	9
Creating a new basic chart.....	10
Saving the chart.....	20
Testing application behavior.....	21
Integrating your development into an application.....	22
Reusing a project in the Designer.....	24
Using More Designer Features.....	25
Using templates.....	27
Creating charts from templates.....	28
Loading data from data sources.....	35
Loading data from a flat file.....	36
Loading data from database (JDBC).....	44
Loading data from In-Memory.....	51
Customizing your chart.....	55
Configuring the chart area.....	56
Configuring chart general properties.....	59
Series graphical representation.....	64
Configuring the legend.....	68

Configuring grids.....	70
Configuring scales.....	72
Configuring 3-D rendering.....	78
Managing decorations.....	81
Managing data style rules.....	87
The Create Style Rule wizard.....	88
Creating rules on data.....	89
Data points.....	90
Data series.....	96
Using the Rules Menu.....	101
Reconfiguring your data source.....	103
Undoing actions.....	106
Printing your chart.....	107
Writing an Application.....	111
Deployment considerations.....	112
Creating a basic Swing application from a JViews Charts component.....	113
XML File Format.....	115
XML Structure.....	117
Schema for XML data file.....	118
Document Type Definition for XML data file.....	121
XML elements.....	122
Interpretation of values.....	129
XML project file.....	130
Next Steps After the Designer.....	135
Integrating the Project File Into an Application.....	136
Integrating the styling into an application.....	137
Styling: CSS versus API.....	138
Index.....	141

Getting Started

Illustrates how to create a basic Cartesian chart using the Designer for JViews Charts.

In this section

Development process

Describes the part of the development process concerning IBM® ILOG® JViews Charts Designer as a flow chart.

Running the Designer

Describes how to run the Designer from recent Microsoft® Windows® or UNIX® operating systems.

Creating a new basic chart

Describes how to create a basic Cartesian chart using the GUI of the Designer.

Saving the chart

Describes how to save a chart.

Testing application behavior

Describes how to test the application.

Integrating your development into an application

Describes how to integrate your chart into an application.

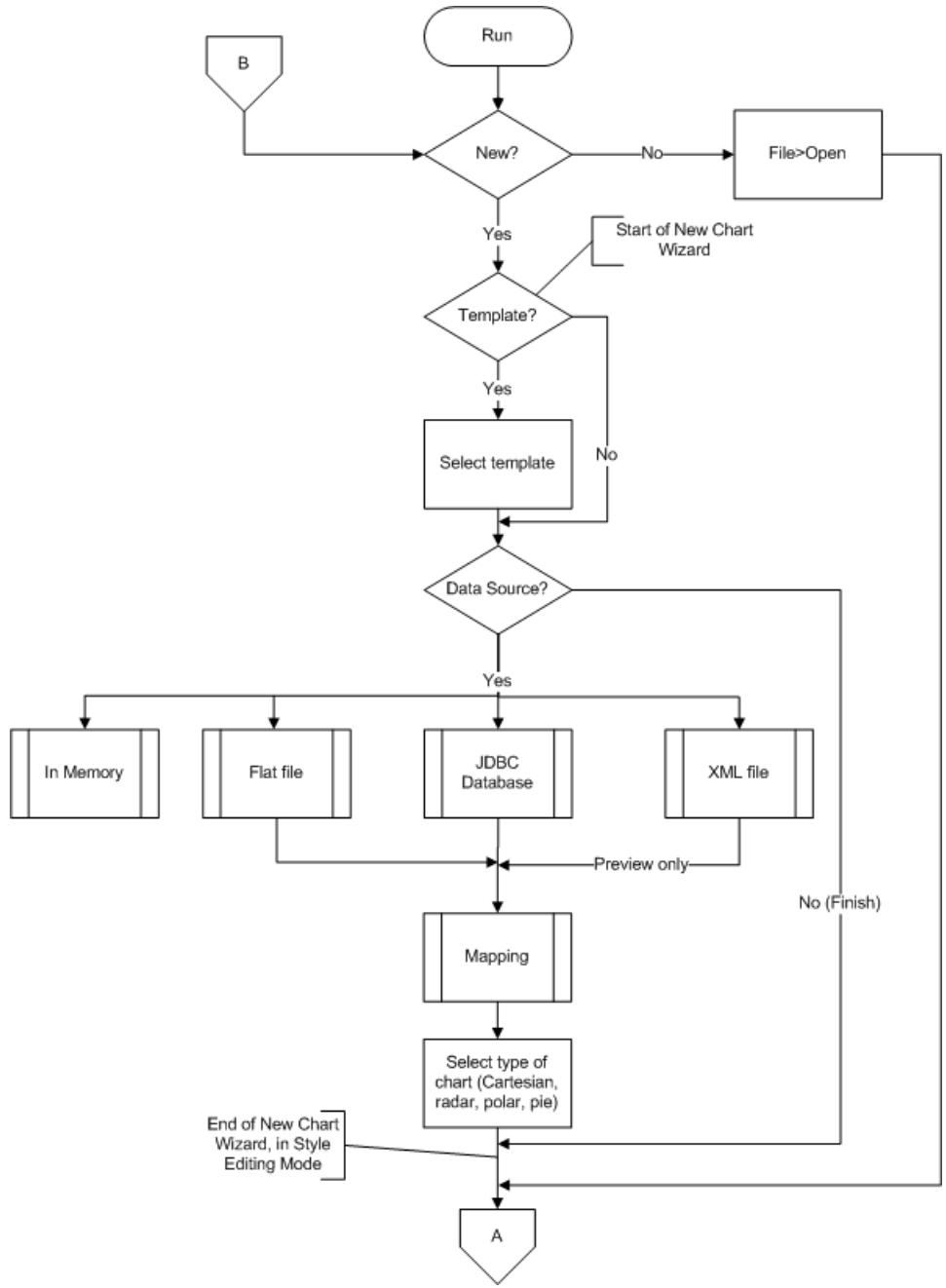
Reusing a project in the Designer

Describes how to refine an existing project in the Designer.

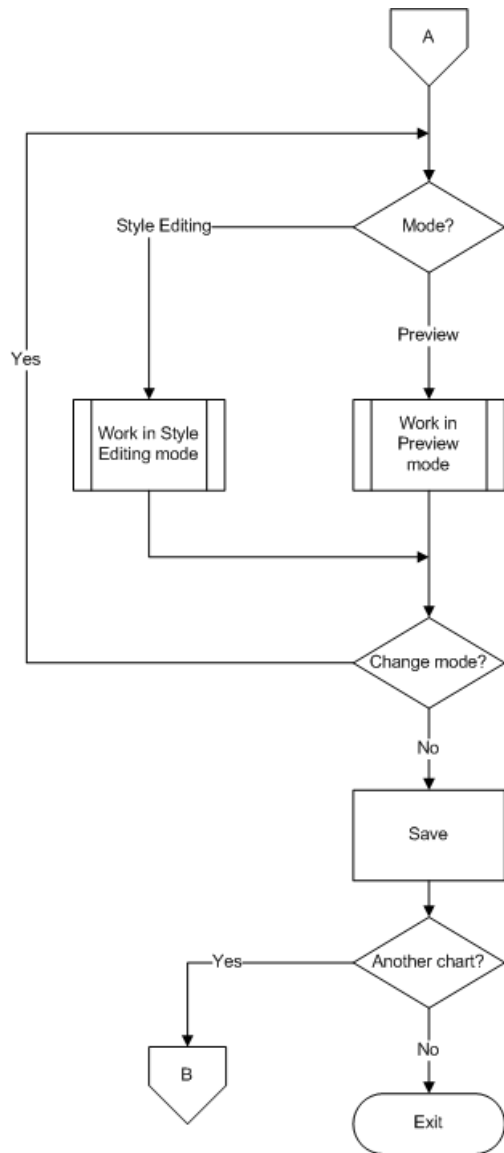
Development process

The IBM® ILOG® JViews Charts Designer gives you a noncode route through the first phase of development.

You must write code to complete your application and deploy it (see *Integrating your development into an application* and *Writing an Application*).



The Development Process Concerning the Designer (1)



The Development Process Concerning the Designer (2)

Running the Designer

The Designer is available in the Start menu. The Designer software is also available in the `<installdir>/jviews-charts85/bin/designer` directory.

To run the Designer from Windows:

1. Navigate to the directory `<installdir>/jviews-charts85/bin/designer`.
2. Double-click `run.bat`.

To run the Designer from UNIX:

1. Make sure your `PATH` environment variable is set correctly to find the directory `<installdir>/jviews-charts85/bin/designer`.
2. Enter `run.sh`.

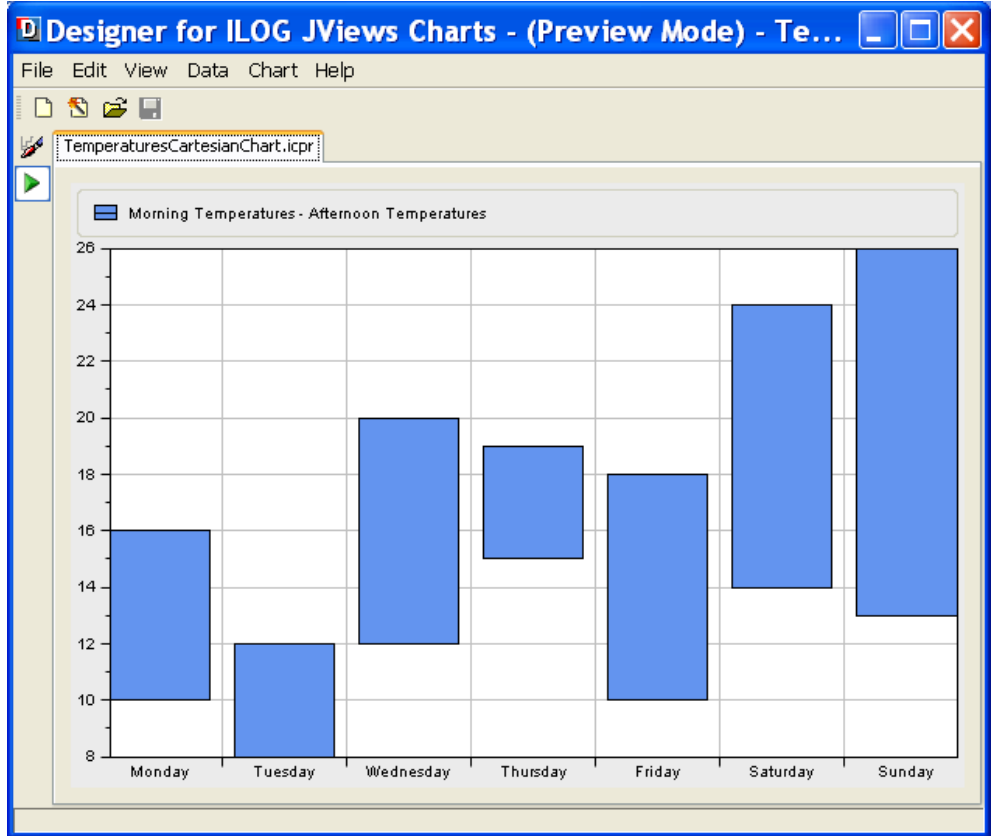
When you start the Designer, it opens with the New Chart Wizard displayed as if you had chosen *File>New from Wizard*.

To set the startup conditions:

1. You can close the wizard if you do not want to create a new chart as proposed.
2. You can also enable or disable the opening of the wizard on startup from its first page.
3. You can change mode using the buttons in the vertical toolbar. Outside the wizard, the way the Designer operates depends on the mode it is in.

Creating a new basic chart

You are going to create a new basic Cartesian chart which displays the morning mean temperatures and the afternoon mean temperatures for each day of the week. The days will be plotted along the abscissa scale and the temperatures along the ordinate scale, as illustrated in *Basic Cartesian Chart*. The example is based on XML data contained in the `temperatures.xml` file.



Basic Cartesian Chart

To create a new chart, use the New Chart Wizard. This wizard may be open on startup. If not, choose *File>New from Wizard*.

In this example you will go through the following steps:

- ◆ *Choosing between template and basic elements*
- ◆ *Choosing the type of data source*
- ◆ *Loading your data*

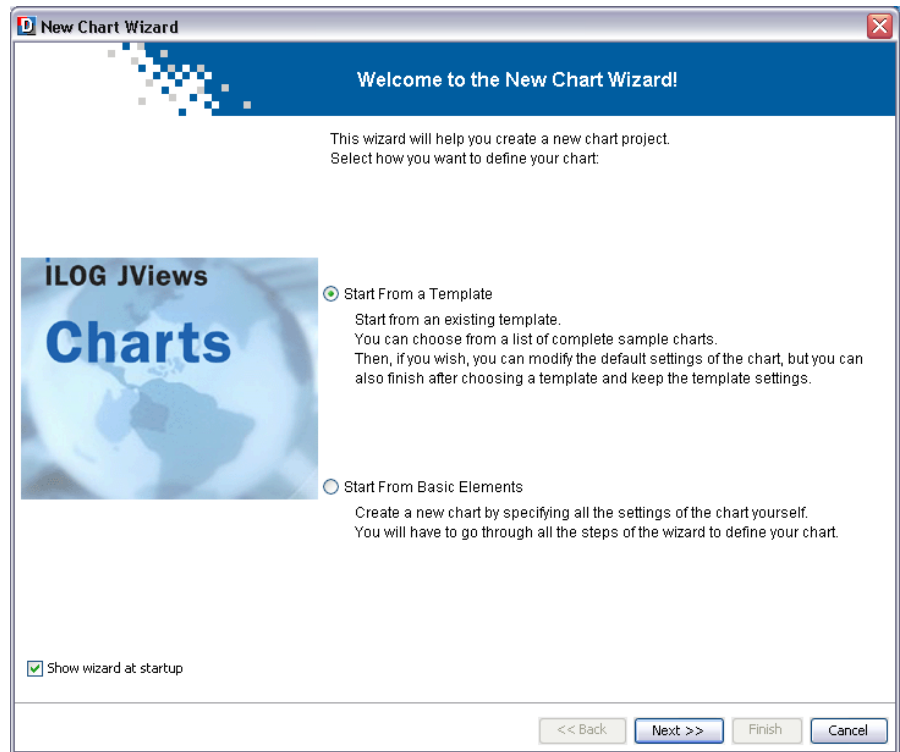
- ◆ *Previewing the data mapping*
- ◆ *Selecting the type of chart*
- ◆ *Choosing the type of graphical representation*
- ◆ *Choosing a theme*
- ◆ *Viewing the final chart*

Choosing between template and basic elements

- ◆ Select the option Start From Basic Elements and click the Next button.

The New Chart Wizard gives you a choice between:

- ◆ Start From a Template
- ◆ Start From Basic Elements



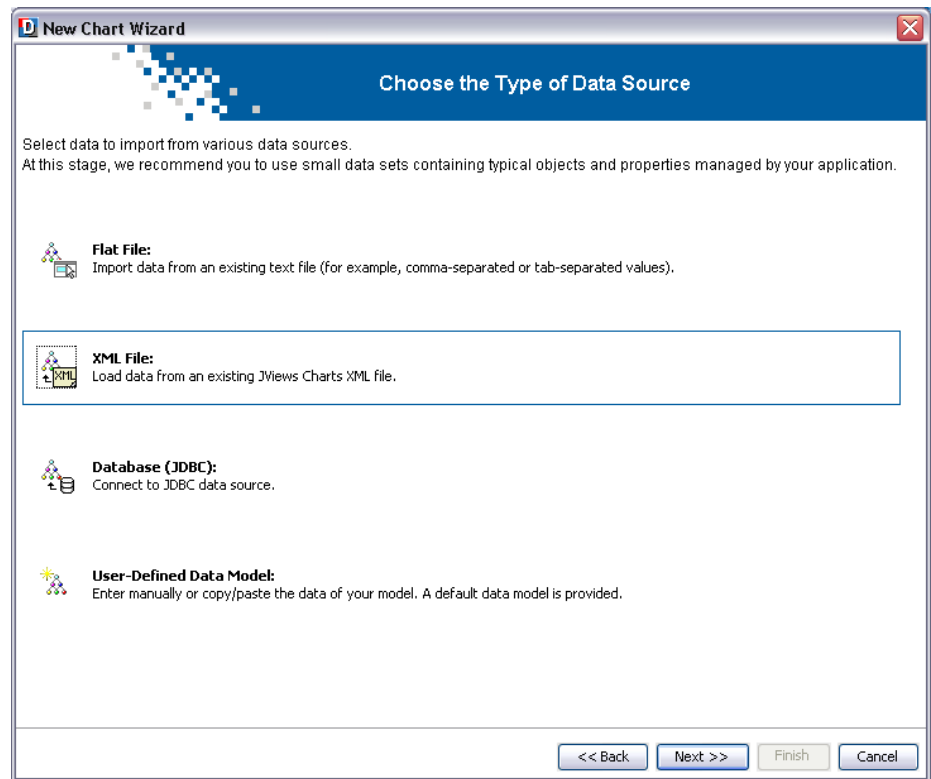
If you want to start from a template, go directly to *Choosing the type of data source*.

Choosing the type of data source

- ◆ Select the option to connect to the XML file and click the Next button.

The New Chart Wizard gives you a choice of the following data sources:

- ◆ Flat file, for example, Excel file in CSV (comma-separated values) format
- ◆ XML file
- ◆ Database supported by JDBC (Java™ Database Connectivity), for example, Microsoft® Access
- ◆ User-Defined Data Model



The example presented in this section is located in:

`<installdir>/jviews-charts86/samples/gallery/data/temperatures.xml`.

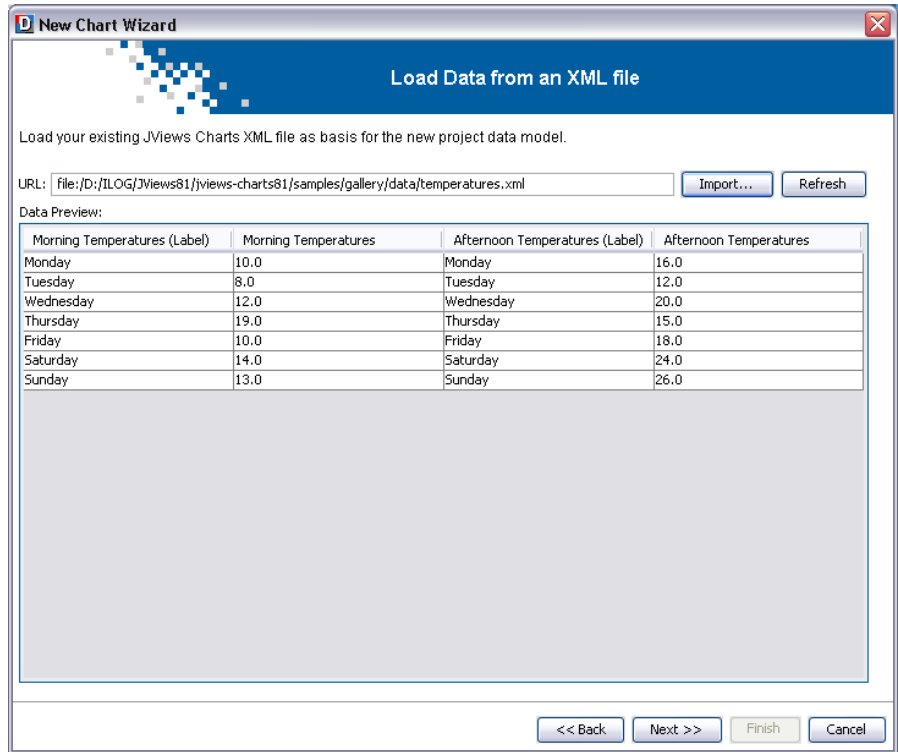
Loading your data

The data to be represented is the morning and afternoon mean temperatures (expressed in degrees Celsius) recorded for each day of the week. The data series is composed of categories.

1. To load your data you can either:
 - ◆ click the Import button at the end of the URL field and browse to the directory containing the `temperatures.xml` file.

or

- ◆ enter in the URL field the path and the name of the file you want to load and click the Refresh button.

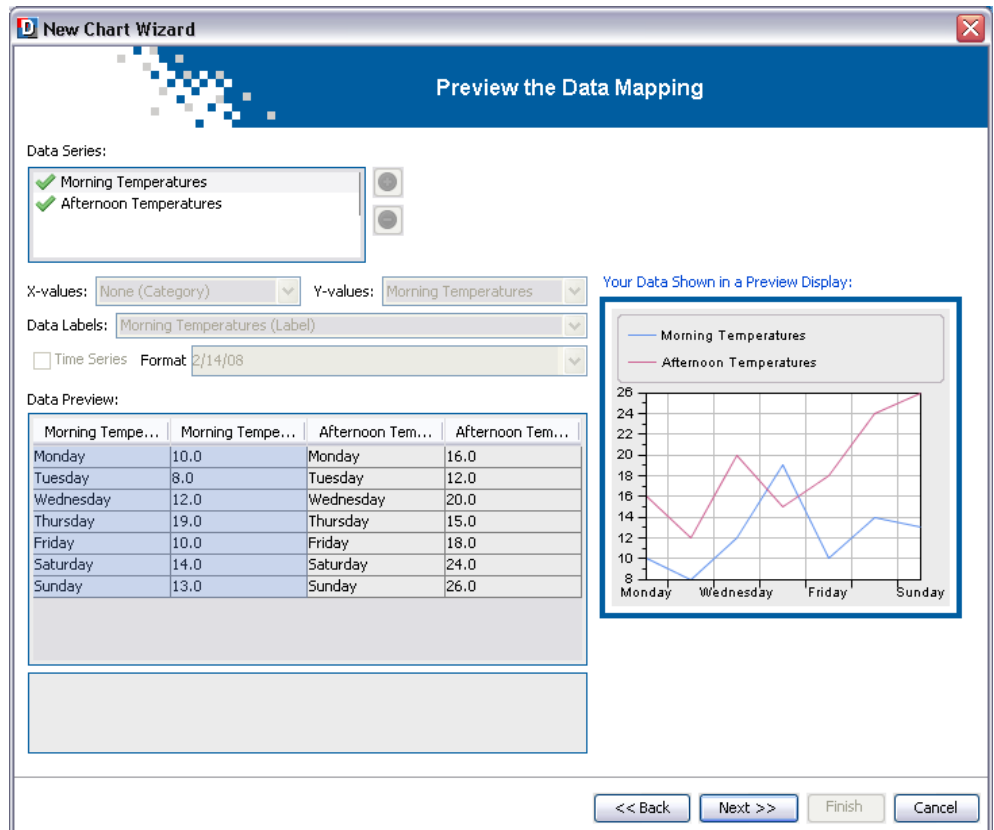


Data contained in the `temperatures.xml` file is displayed in the Data Preview area.

2. Click Next to continue.

Previewing the data mapping

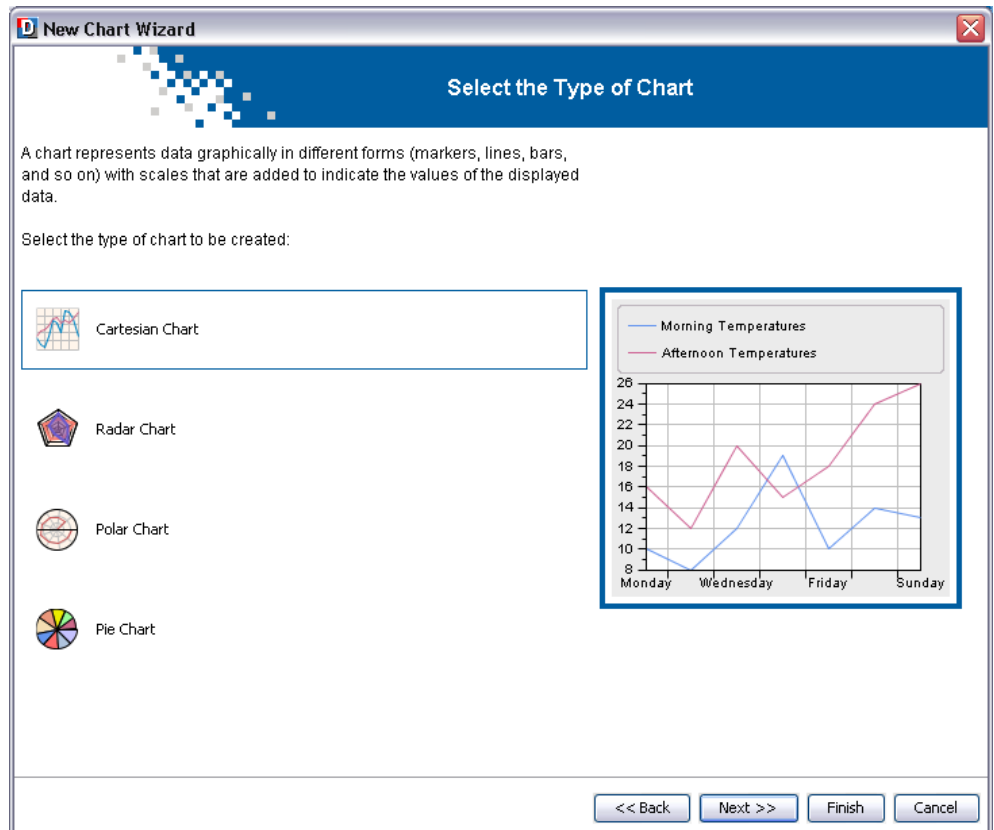
In this example, the data is already mapped. This window presents the data mapping as it is defined in the `temperatures.xml` file.



1. Click one of the series in the Data Series table to see the corresponding mapping in the Data Preview table.
2. Click Next to continue.

Selecting the type of chart

At this step you define the type of chart you want to create.

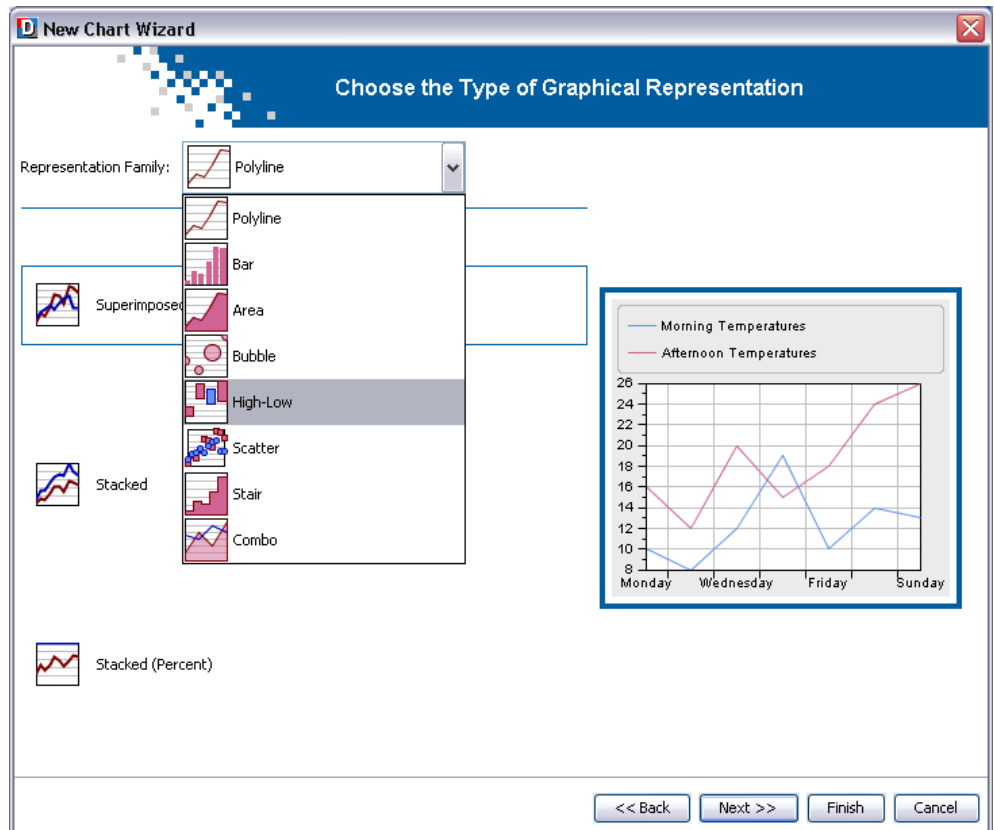


It is possible to create four different types of chart: Cartesian, Radar, Polar and Pie. To choose one of these types, click the corresponding icon.

1. Select the option Cartesian Chart.
2. Click Next to continue.

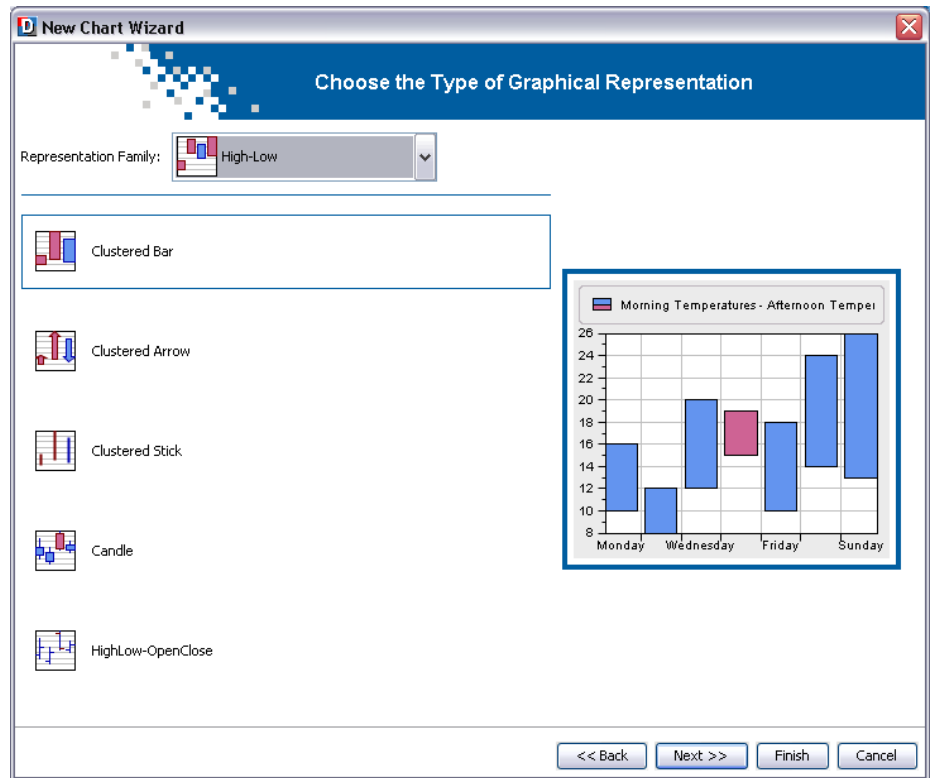
Choosing the type of graphical representation


You are going to display the two sets of temperatures with two polylines and with a High-Low bar representation, which represents both the data sets of the morning and afternoon mean temperatures.



The first polyline represents the data set of the morning mean temperatures and corresponds to the first series of values in the data source. The second polyline represents the data set of the afternoon mean temperatures, which corresponds to the second series of values in the data source.

1. Choose High-Low from the Representation Family combo-box.

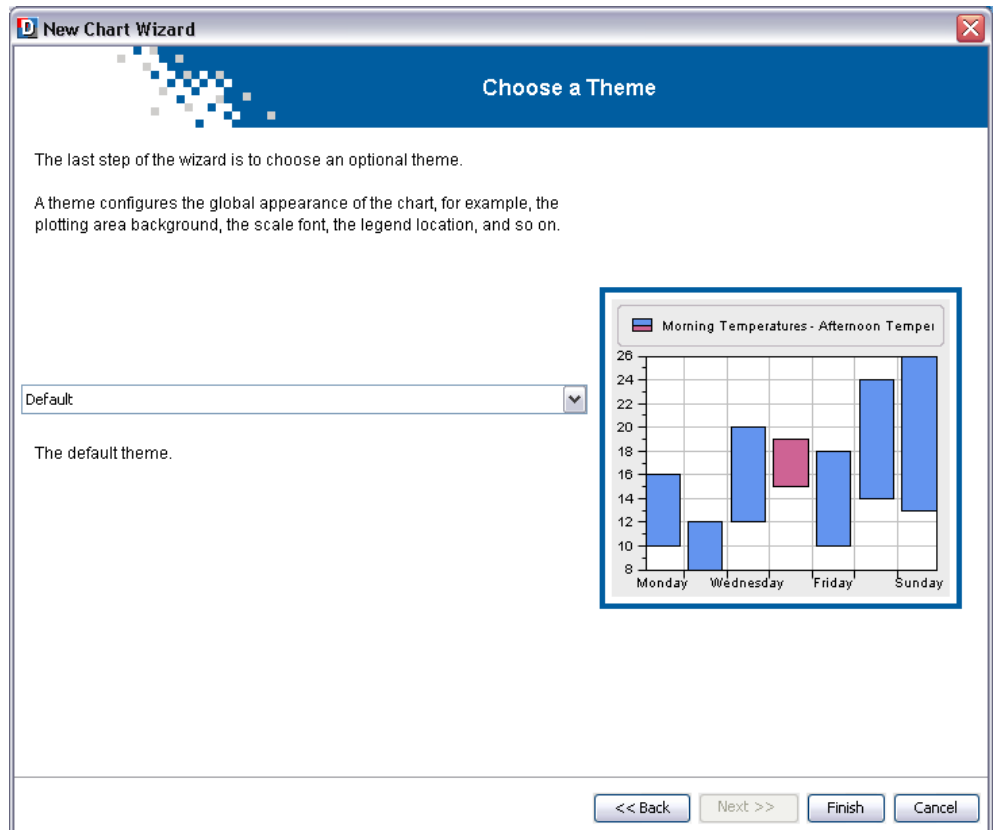


2. Select the Clustered Bar  option and click the Next button to continue.

Choosing a theme

A theme consists only of a style sheet. It is a convenient way of applying a common look that you want to reuse in different projects.

The default themes provided with the Designer only contain static styling rules (no data styling) and do not change the setting you could have set in the previous steps.

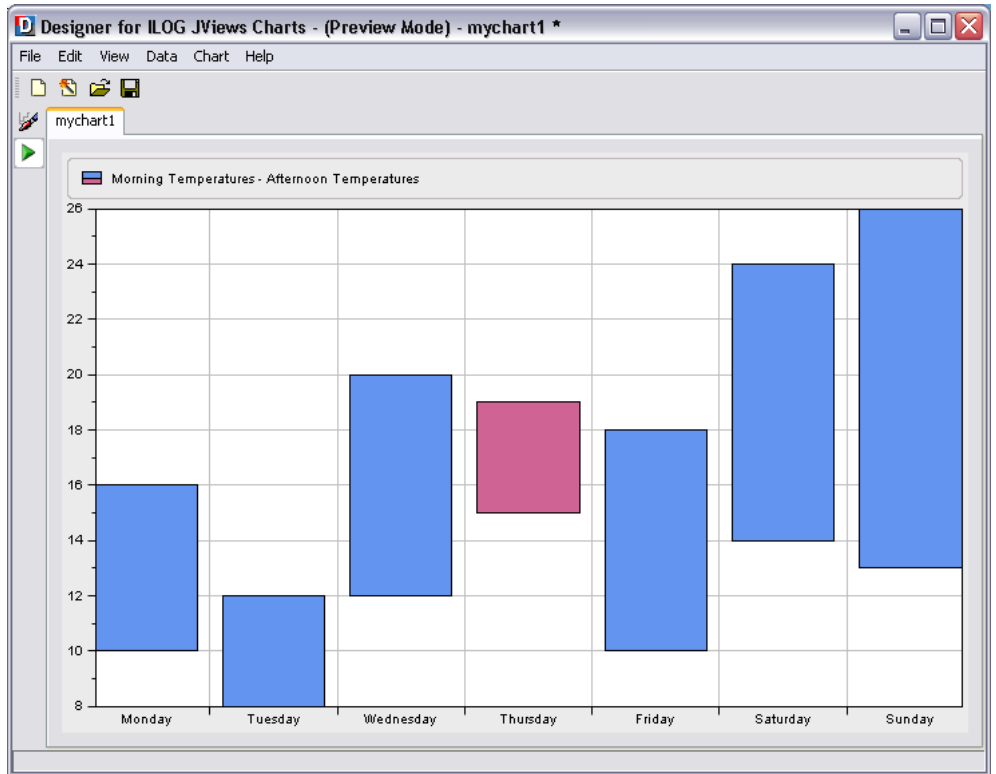


- ◆ Leave the Default theme and click the Finish button.

This was the last step of the Wizard.

Viewing the final chart

Here is how the Cartesian chart should appear:



Saving the chart

It is a good idea to save your charts periodically as you create them.

To save a chart for the first time:

1. Make sure you have already created a directory in which your charts will be saved.
2. Choose Save As from the File menu.

A file selection dialog box appears.


3. Select the directory in which you want to save the chart.
4. Type the name of the chart. For this example type `TemperaturesCartesianChart.icpr`.

You should use the `.icpr` filename extension, which stands for JViews Charts Project.

5. Click Save in the dialog box.

The chart is saved with the filename you have just typed.

To save an edited chart:


- ◆ Click the Save button  from the toolbar or choose Save from the File menu to save an edited chart, that is, one that has already been saved and that you have opened and edited.

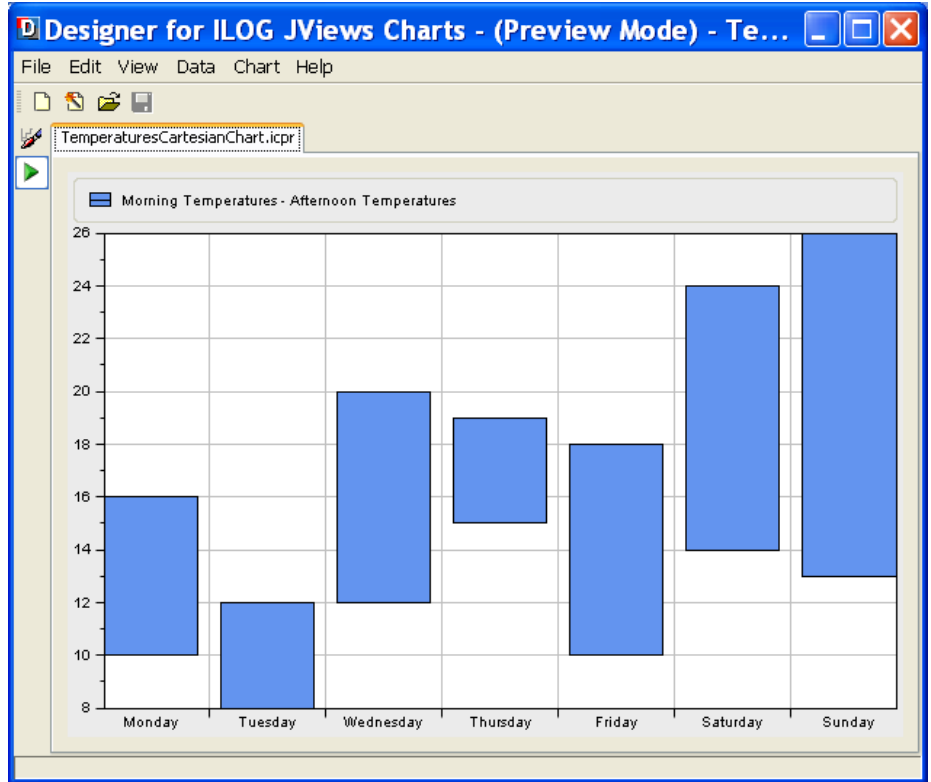
The updated chart is saved in the same place with the same name.

Once you have saved your chart, you might want to integrate it in your application. For more information, refer to *Integrating your development into an application*.

Testing application behavior

By default, in the Designer you are in Style Editing Mode. In Style Editing Mode, you can define style rules.

- ◆ To switch to Preview Mode, click the Preview button  in the vertical toolbar. Preview Mode shows you the chart as it will be in an application. This mode allows you to test the application behavior.



Integrating your development into an application

You can integrate a complete chart into an application through the project file or you can integrate only the style sheet to have the styling developed in the Designer applied to real data in an application.

To integrate a chart into a Java application or applet:

1. Make sure all the project files (the `.icpr` and the `.css` files) are accessible from the application.
2. In the source code of your application, add the following lines of code once the chart instance has been created:

```
IlvChart chart = ...;
try {
    URL project = ...;
    // Apply the project file to the chart chart.setProject(project);
} catch (IOException e) {
    ...;
} catch (IlvStylingException se) {
    ...;
}
```

The chart is thus configured according to the settings of the project file.

The code for creating the chart instance and loading the project file is shown in:

```
<installdir>/jviews-charts86/codefragments/chart/project-viewer/src/
ProjectViewer.java.
```

Note: A project file includes both data and style sheet. When a project is set, it replaces any data source and style sheets that may have been previously set on the chart.

For details on how to integrate the chart into a thin-client application, see The JViews Charts Faces component set in *Building Web Applications*.

For details on how to integrate the chart into a Rich Web Client application, see Add a Rich Web Charts to a JSP page in *Building Web Applications*.

If you have a custom data model and your application is already defined, you can use the Designer for styling only on a sample of dummy data or on an XML dump of your data.

Then you can integrate the style sheet from the Designer into your application. You can apply the styling defined in the style sheet to the real data used in the application. The Designer makes it much easier to write the style sheet.

To style the chart component of your Java application, applet or DHTML thin-client application using an existing project style:

1. Make sure that the project styling file (`.css` file) is accessible from the application.

2. In the application, after creating the chart and loading the data model, use the following code:

```
try {
    chart.setStyleSheets(new String[]{"yourStylingFile.css"});
} catch (IlvStylingException x) {
    System.err.println("Cannot load style sheets: " + x.getMessage());
}
```

For more information on how to load style sheets in a Java application, applet or DHTML thin-client application, see *Styling in Developing with the SDK*.

You can also look at the Charts CSS sample that can be found in:

```
<installdir>/jviews-charts85/samples/css
```

For more information on how to integrate a style sheet into a DHTML-based JSF thin-client application, see *Styling chart data with CSS in Building Web Applications*.

For more information on how to integrate a style sheet into a Rich Web Client application, see *Add a Rich Web Charts to a JSP page in Building Web Applications*.

Reusing a project in the Designer

You may want to refine a prototype or make changes to the styling and the Designer offers a quick and easy way of doing this.

You can only reuse projects in the Designer that have been developed exclusively with the Designer. Projects that have been further developed in the SDK are likely to contain features that are not available in the Designer.

- ◆ Drag a project file from a directory into a Designer window. The chart configured by the project file opens automatically in the Designer.

Using More Designer Features

Describes how to handle charts through the Designer user interface. In particular, presents examples of various Designer features not covered in the Getting Started.

In this section

Using templates

Explains how to create charts by using a template as a starting point.

Loading data from data sources

Describes how to create a chart by loading new data from various data sources.

Customizing your chart

Describes how to customize an existing chart.

Managing data style rules

Explains how to use the Style Rule Wizard to specify rendering attributes for data series and single data points.

Using the Rules Menu

Explains how to use the Rules Menu within the Designer to manage your rules.

Reconfiguring your data source

Explains how to change the data source type from XML to in-memory.

Undoing actions

Describes the actions to which the Undo command can be applied.

Printing your chart

Explains how to print your chart from the Designer.

Using templates

Explains how to create charts by using a template as a starting point.

In this section

Creating charts from templates

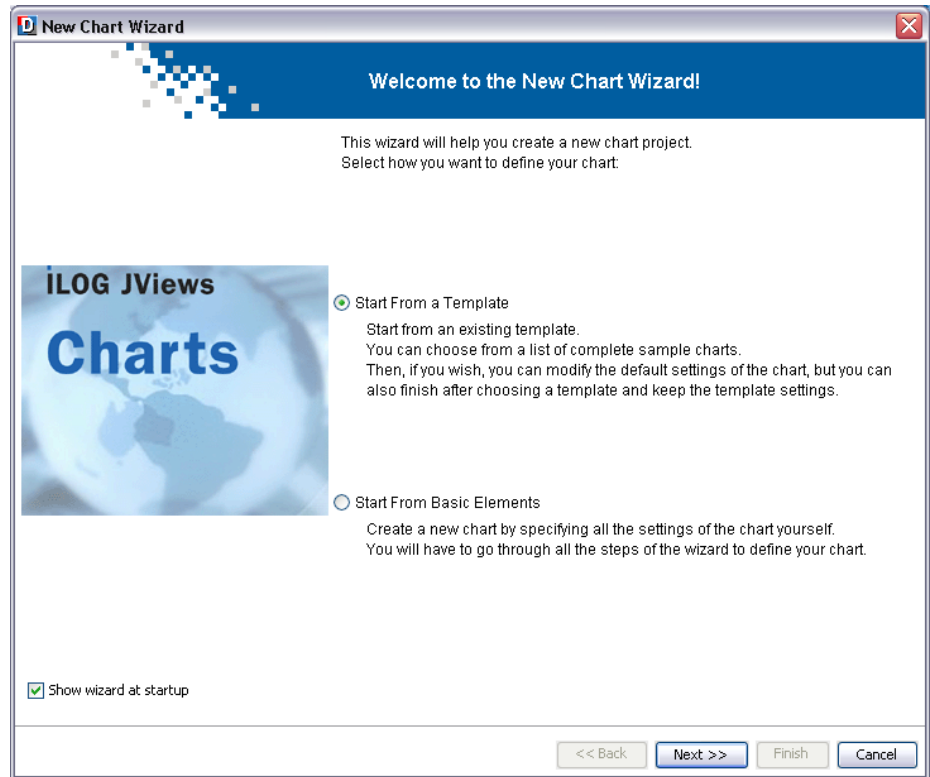
Describes how to create charts, customize them and interact with these charts in different manners.

Creating charts from templates

Within the New Chart Wizard, you can use templates for creating your chart. A template is a predefined project which includes a style sheet designed for a specific type of application or style of presentation and an appropriate data model.

To start a new chart based on a supplied template:

1. Choose *File>New from Wizard*. The New Chart Wizard starts.



2. Select the option to start from a template and click Next.

The available templates are as follows:

◆ 3D Vertical Bar

A chart displaying 3 series using 3D vertical bars ordered by category.

◆ 3D Pie

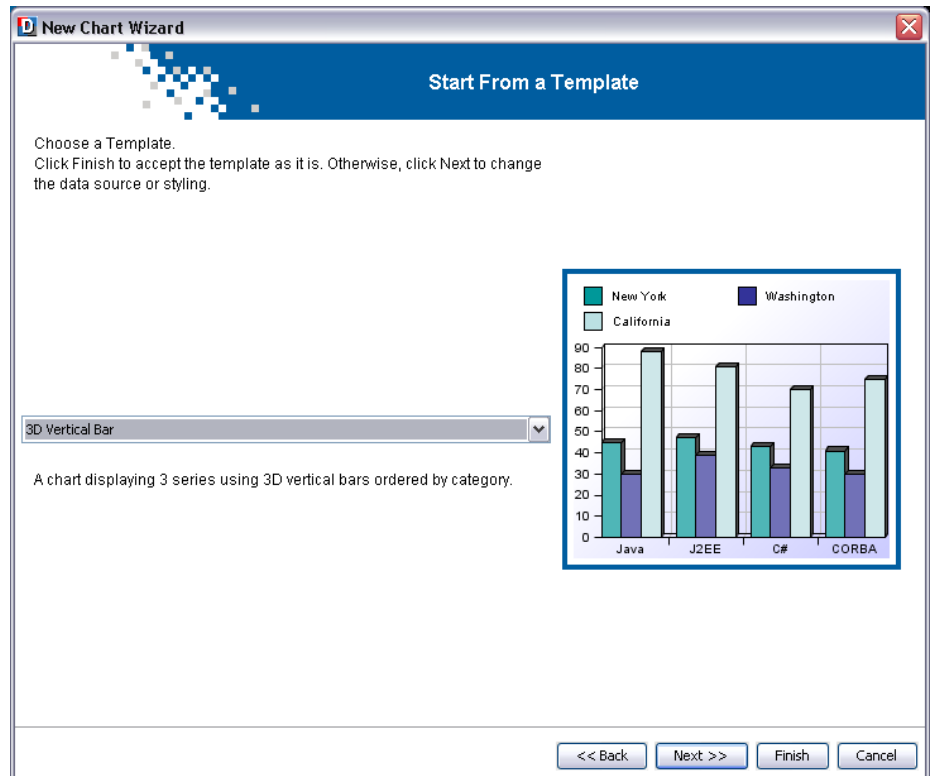
A 3D pie chart displaying one series composed of 5 values. The value of each point is displayed as a percentage over each slice.

◆ Multiple Representation

A chart mixing different representation types: a Bar, a Polyline, a Area and a scatter renderers.

- ◆ Superimposed Polylines
A Cartesian chart with polylines.
- ◆ Superimposed Vertical Bars
A Cartesian chart with bars.
- ◆ High-Low Bars
A High-Low-Open-Close chart.
- ◆ Clustered Bars in a Radar Chart
A radar chart with bars.

3. From the list of templates, select 3D Vertical Bar.



From here, you can proceed in two different ways:

- ◆ keep the predefined template as is,

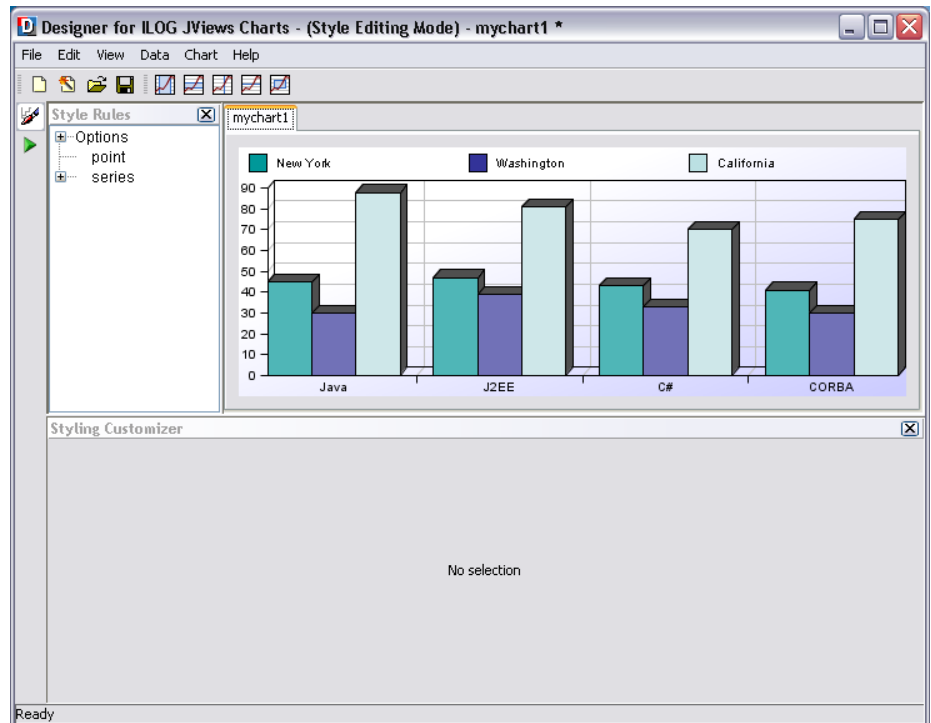
or

- ◆ customize the template by changing the data source and the styling.

To keep the predefined template:

- ◆ Click Finish to exit the Wizard if you are happy with the predefined template you have chosen from the list.

Your chart appears in the Designer with the style and data defined by the template you have selected. In this example you have selected the template “3D Vertical Bar”, your chart displays three series using 3D vertical bars ordered by category, as follows:

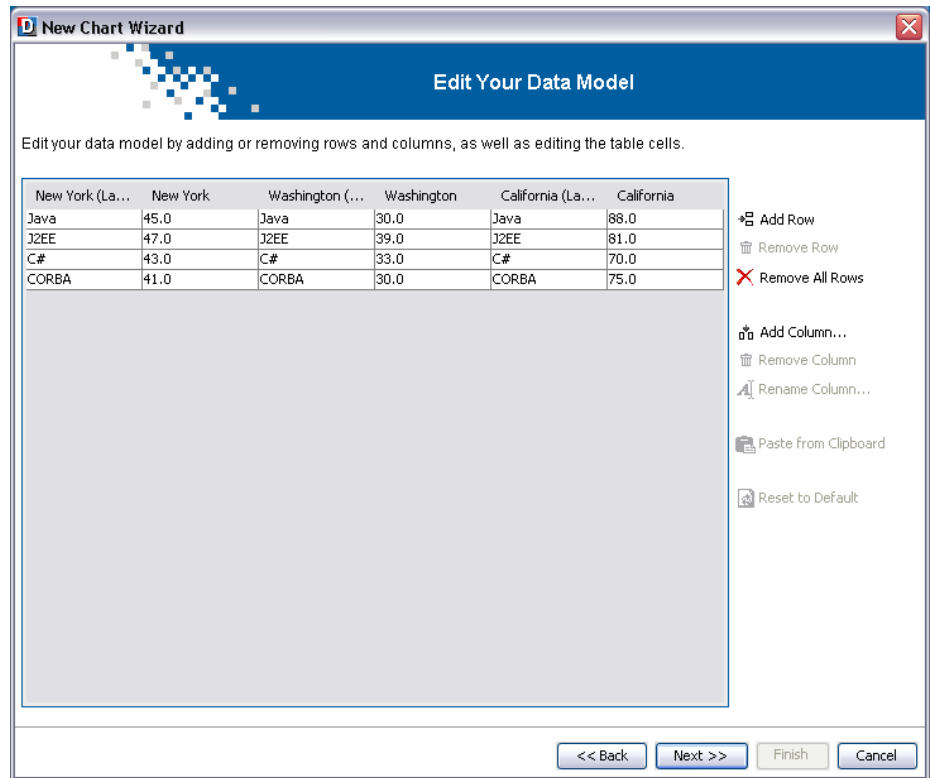


To customize the template by changing its data:

1. Select the template and click Next to continue.

The window Choose the Type of Data Source opens.

The Wizard automatically proposes you the data source appropriate for the template you want to customize. In this example, the Wizard proposes the User-Defined Data Model option. Click Next to continue.



2. Edit the data model by adding or removing row and columns, clearing all the data or resetting the model to the default value.

To customize the template by changing its styling:

- ◆ Redefine the type of chart and the graphical representation. For more details on how to do this, refer to *Creating a new basic chart*.

If you want to create a family of charts with the same data and based on the same styles, you may wish to create your own template.

To create a new template:

1. Create a chart that contains the required data, with all the styling you need.

You can use the `TemperaturesCartesianChart.icpr` that you created in the *Getting Started* and then apply the customizations proposed in *Customizing your chart*.

You need the following files:

- ◆ a JViews Charts Project file (`.icpr`)
- ◆ a style sheet file (`.css`)
- ◆ a property file (`.properties`)

It is important to have the three files in the same location.

From the menu bar, choose *File>Save As* and save the project file in the <installdir>/templates directory.

2. The style sheet.

The .css file is automatically created when you create your project. It has the same name (TemperaturesCartesianChart.css) and is placed in the same directory.

3. Create a .properties file with the short and long descriptions of your template.

The template properties file must comply with a specific syntax so that a custom template is properly integrated into the wizard. The expected syntax for the .properties file is shown in the following extract:

```
ShortDescription = 3D Vertical Bar
LongDescription = A chart displaying 3 series using 3D vertical bars
ordered by category
```

The properties file must contain the following keys:

◆ . ShortDescription

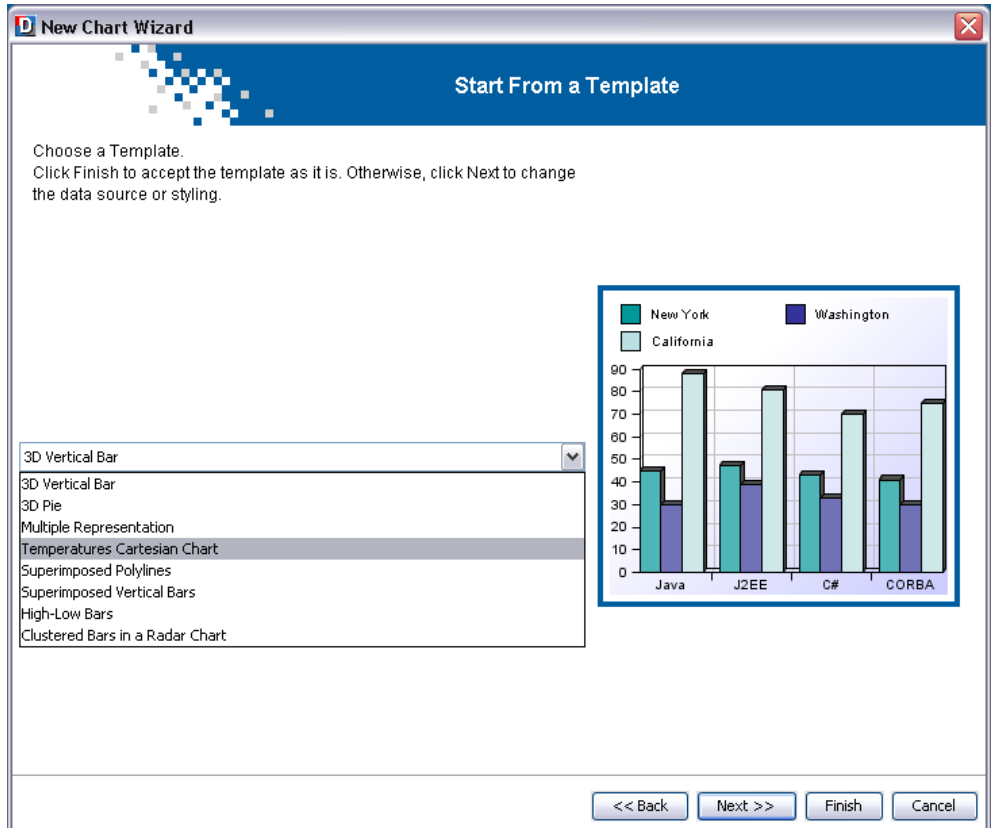
The text corresponding to this key is displayed in the combo box that lists all the templates.

◆ . LongDescription

The text corresponding to this key is displayed below the combo box in a text area to provide a detailed description of the template.

Save the file with same name of your project, TemperaturesCartesianChart.properties and place it in the <installdir>/templates directory.

After you have saved this set of files, your template appears with the predefined templates in the New Chart Wizard.



Loading data from data sources

Describes how to create a chart by loading new data from various data sources.

In this section

Loading data from a flat file

Describes how to load data from an existing text file (comma-separated or tab-separated values).

Loading data from database (JDBC)

Explains how to retrieve data stored in a database.

Loading data from In-Memory

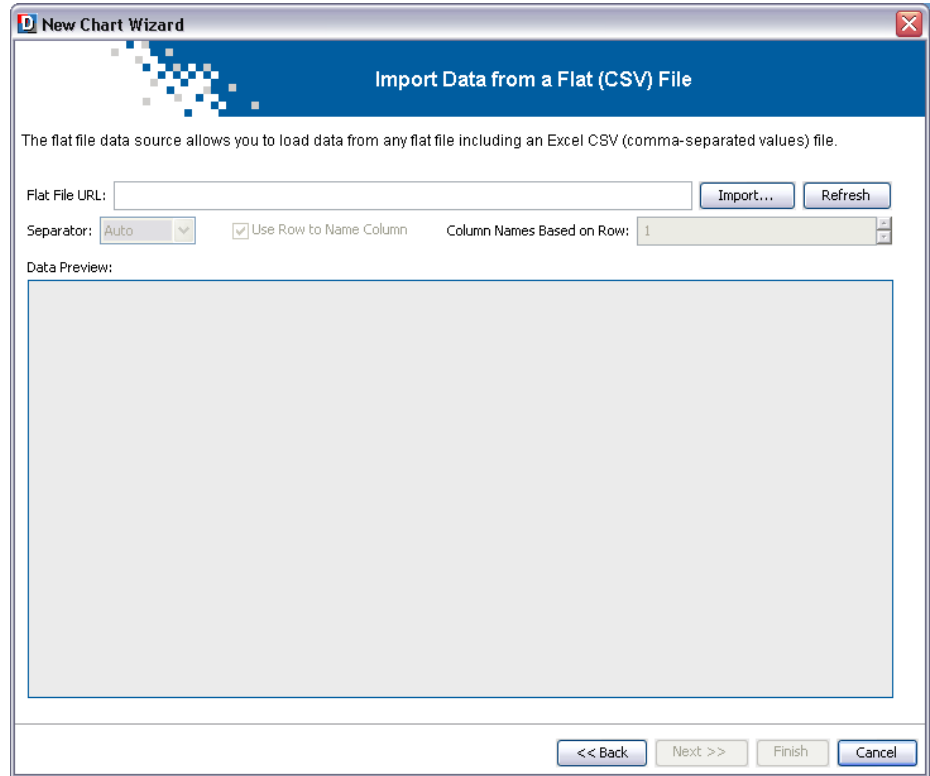
Explains how to retrieve data from in-memory data source.

Loading data from a flat file

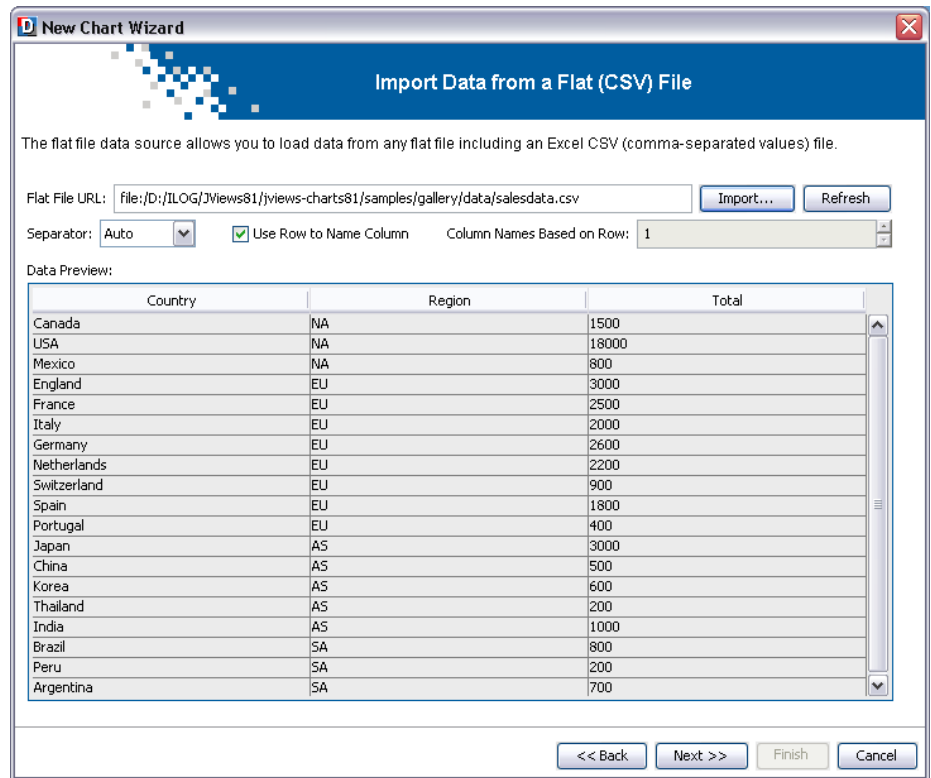
For more information on the flat file data source, see *Populating the chart* in *Introducing JViews Charts*

Loading data from a flat file

1. Choose File>New from Wizard.
2. Select Start From Basic Elements and then the flat file option.



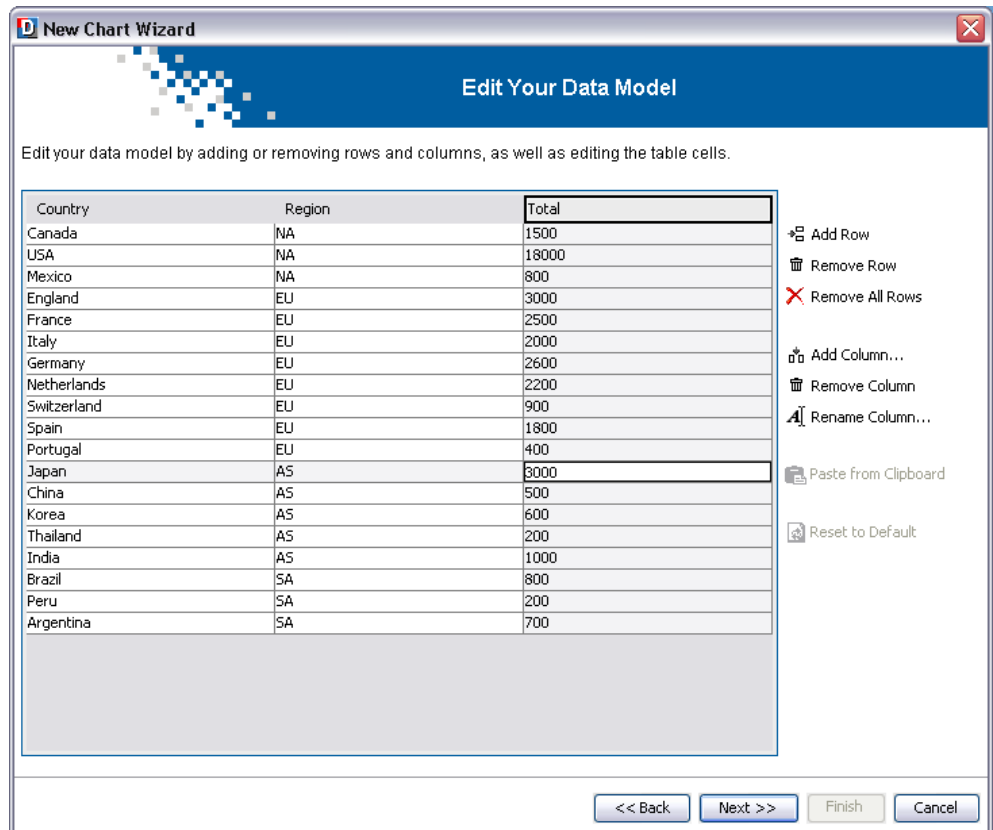
3. Browse to `salesdata.csv` file and click Open.



4. Click Next to continue.

Editing your data model

You are going to edit the data model by removing the column Region.



1. Select the column Region by clicking one of its cells and then click the Remove Column button.
2. Click Next to continue.

Defining the data mapping

The data mapping defines which columns of the table should be used for the display of the chart and in which role. The data mapping is an association between the columns of the table and a list of data series. Each data series is composed of the following elements:


- ◆ a column of Y values (mandatory)
- ◆ a column of X values (optional)
- ◆ a column of data labels (optional)

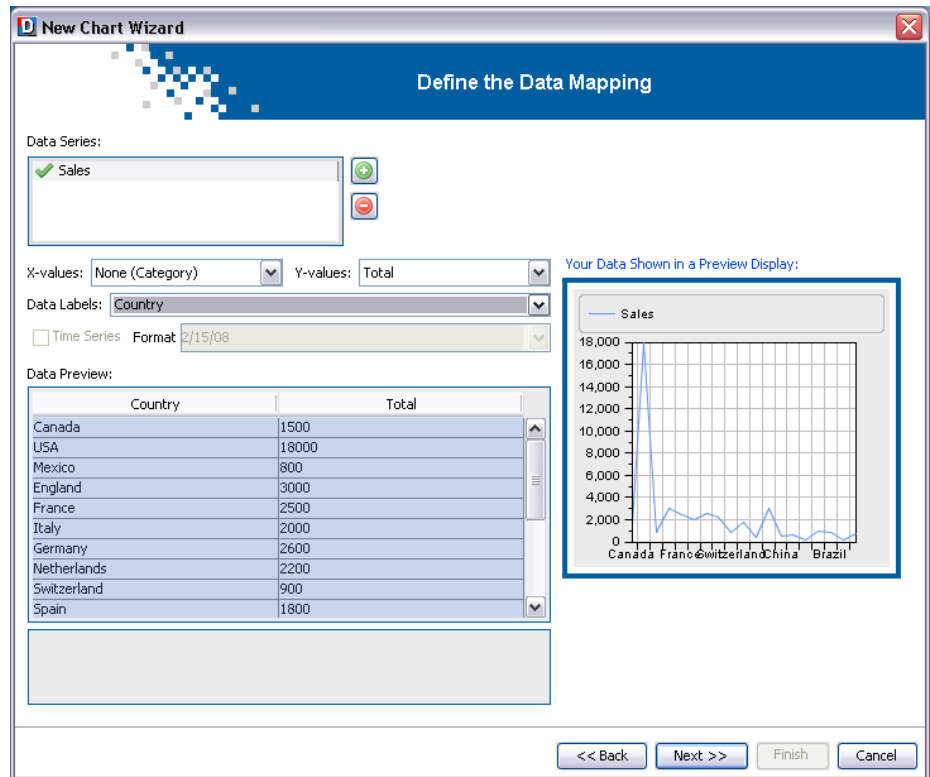
The order of the data series is important: the data series are drawn in the same order, so that the last data series appears on top of the others.

To create a new series you have to click the  button, rename the data series newly created and choose the table column(s) to which it corresponds. For the X and Y values of a data

series you can only choose a numerical column; for the labels you can use a non-numerical column.

The data mapping is complete when all the columns containing the data that you want to view are mapped to data series. You are going to add a new series and call it Sales.

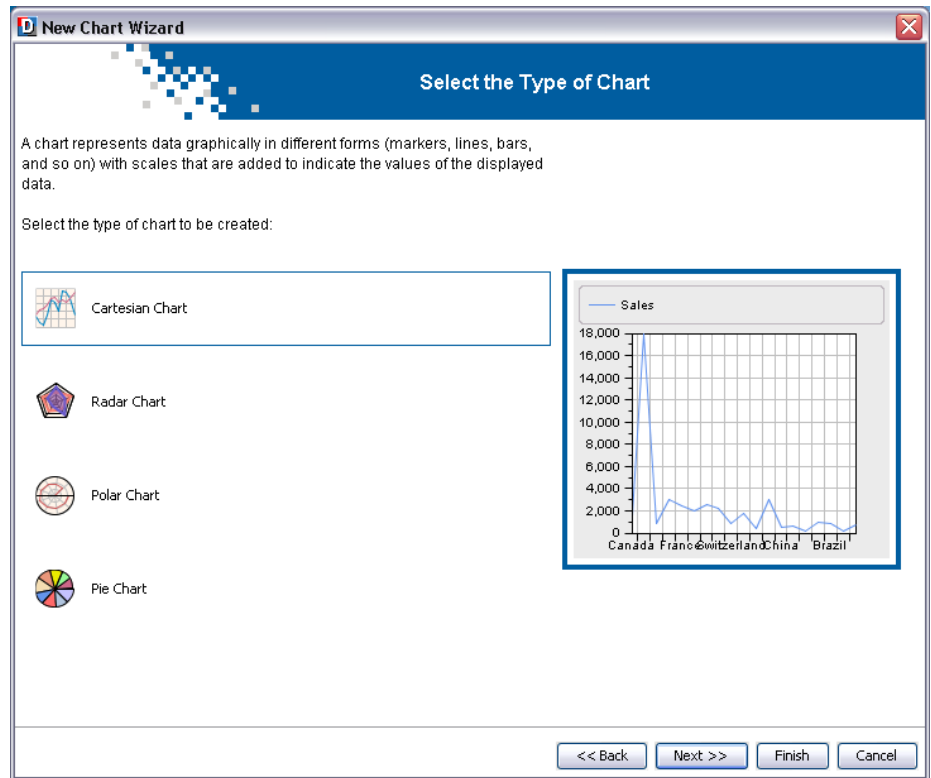
1. To add a new series click the the  button.
2. Double-click the "New Series 1" in the Data Series field, rename it Sales and press Enter to validate.
3. In the Y-values field select Total.
4. In the Data Labels field select Country.



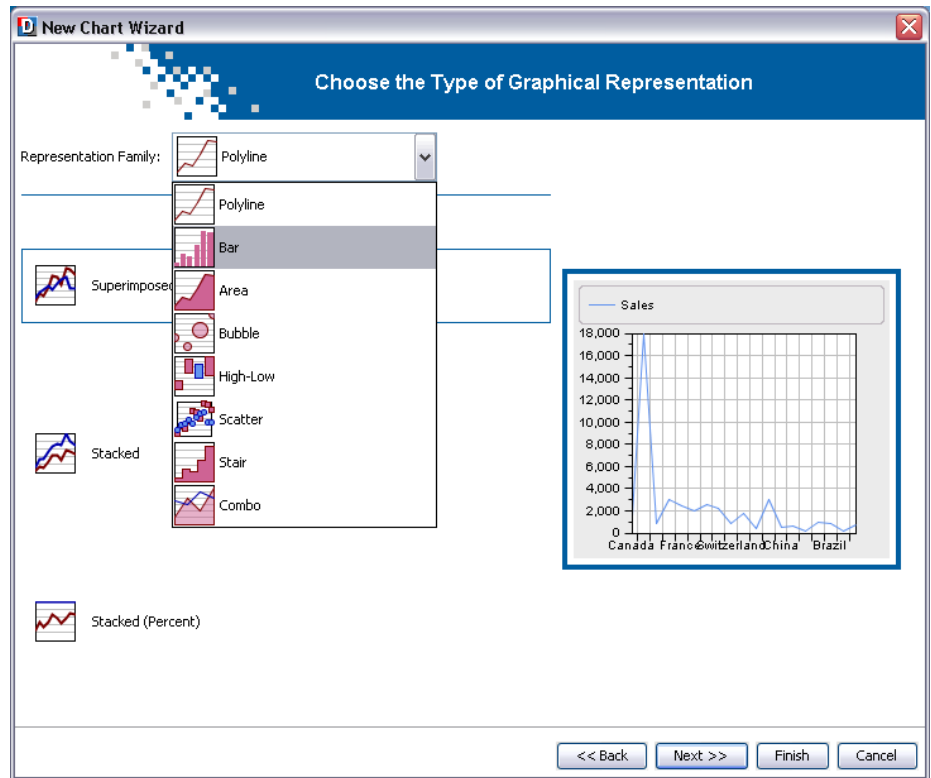
5. Click Next to continue.

Selecting the type of chart

1. Create a Cartesian chart.



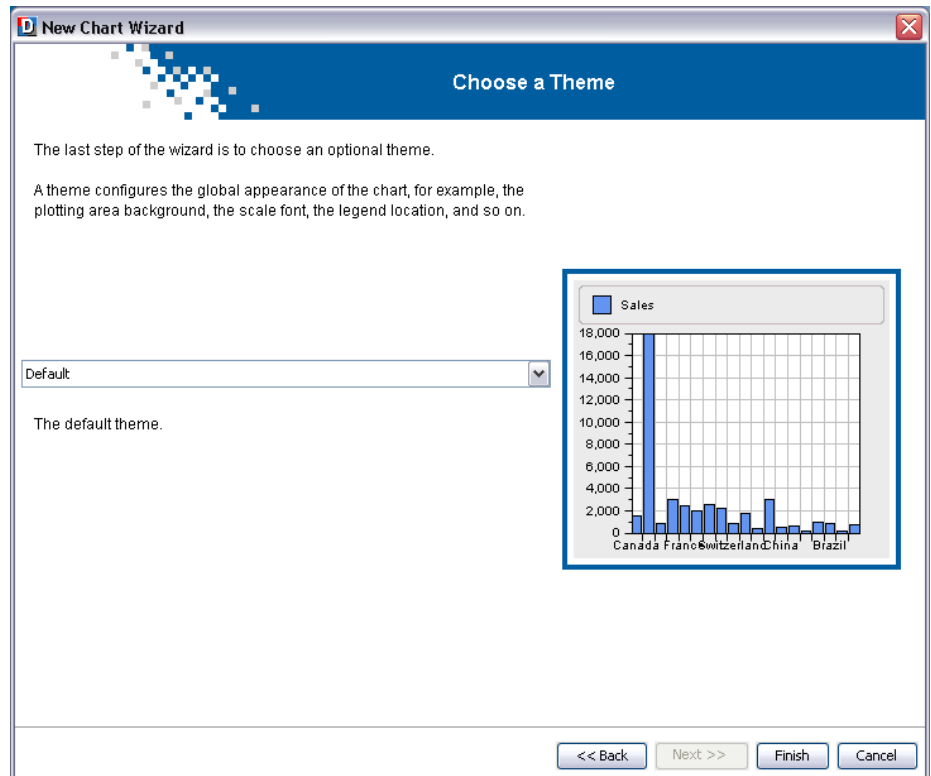
2. Select the option Create a Cartesian chart and click Next.
3. From the Representation Family list, select the option Bar.



4. Select the option Clustered and click Next.

Choosing a theme

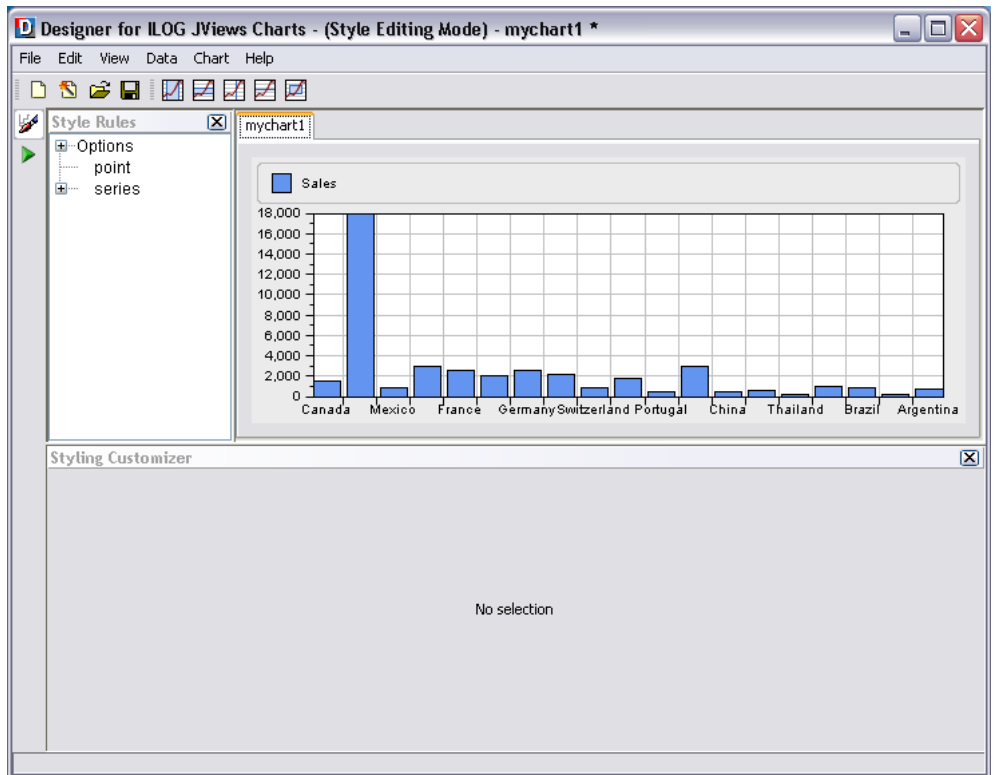
1. For this example, you can leave the Default theme.



2. Click Finish to end the wizard session.

If you need to make changes, you can click Back instead to return to specific wizard pages.

The resulting chart should appear in the Designer, as follows:



Loading data from database (JDBC)

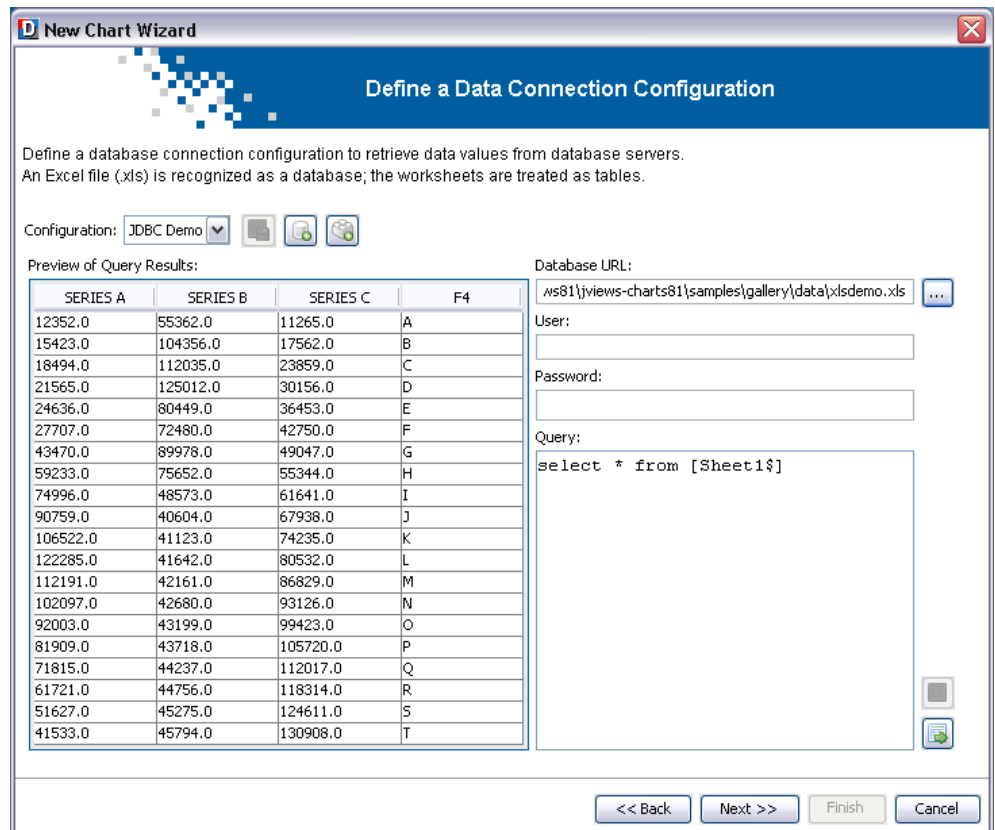
IBM® ILOG® JViews Charts supports the use of the JDBC interface to retrieve data value from database servers. For more information on the database data source, see Populating the chart. in *Introducing JViews Charts*. You can also find more information about JDBC on the JavaSoft site at: <http://java.sun.com/products/jdbc>.

Loading data from a database

1. Choose *File>New from Wizard*.
2. Select Start From Basic Elements and then the Database (JDBC) option.

Defining a data connection configuration

The first page allows you to define the database data source.



1. You must specify the following:
 - ◆ A URL to connect to the database.

- ◆ Your user name and password, if these security items are required.
- ◆ The SQL query string.

URL

The Universal Resource Locator (URL) can indicate a connection to a remote database or a path to a local file. In this example, the database is an Excel file named `xlsdemo.xls`.

User Name and Password

If your data is in a local file, you will not need to enter a user name and password. If your data is in a database, the Database Administrator can provide the information.

SQL Query

You must enter a query to retrieve the data from the database. Initially, a default query is shown; this query selects all columns from a sheet, as follows:

```
SELECT * FROM [Sheet1$]
```

2. Modify the query to retrieve data from Sheet2.

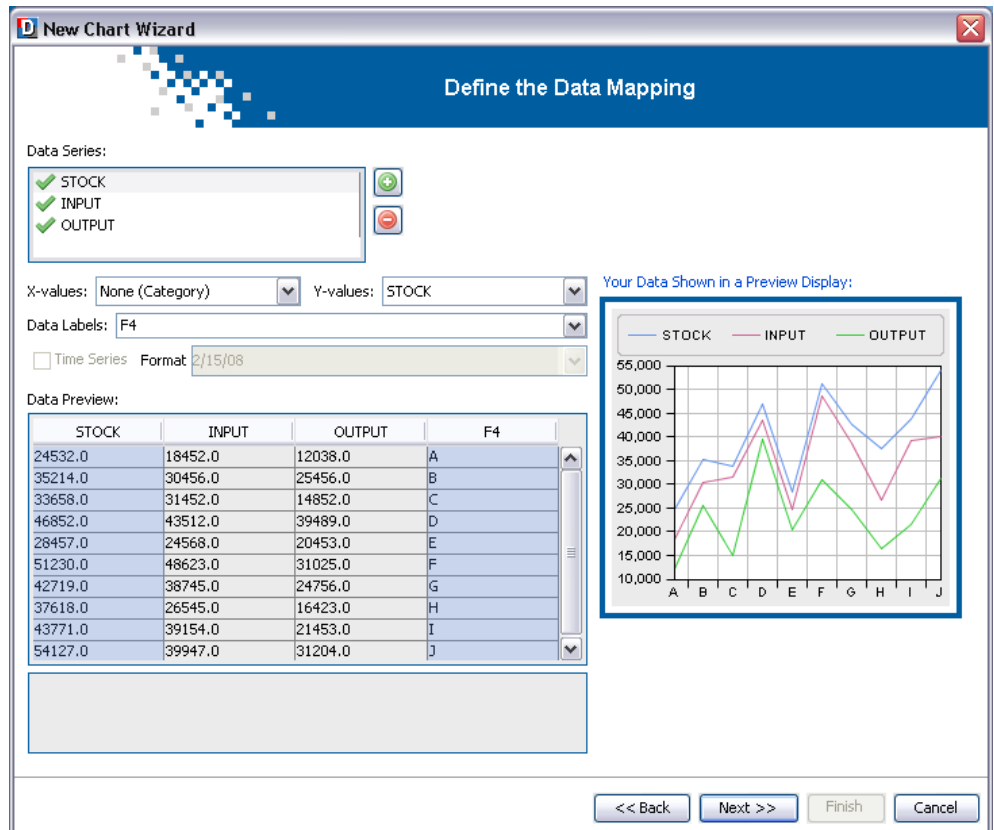
In the Query field, replace Sheet1 by Sheet2 and click the button Execute the Query. You will see the data in tabular form in the Preview pane.

STOCK	INPUT	OUTPUT	F4
24532.0	18452.0	12038.0	A
35214.0	30456.0	25456.0	B
33658.0	31452.0	14852.0	C
46852.0	43512.0	39489.0	D
28457.0	24568.0	20453.0	E
51230.0	48623.0	31025.0	F
42719.0	38745.0	24756.0	G
37618.0	26545.0	16423.0	H
43771.0	39154.0	21453.0	I
54127.0	39947.0	31204.0	J

Check that the data retrieved is the required one and click Next to continue.

Defining the data mapping


This page allows you to specify the mapping of the database columns to graphic properties.



The data mapping defines which columns of the query results should be used for the display of the chart and in which role. The data mapping is an association between the columns of the query results and a list of data series. Each data series is composed of the following elements:

- ◆ a column of Y values (mandatory)
- ◆ a column of X values (optional)
- ◆ a column of data labels (optional)

The order of the data series is important: data series are drawn in the same order, so that the last data series appears on top of the others.

To create a new series you have to click the  button, rename the data series newly created and choose the query results column(s) to which it corresponds. For the X and Y values of a data series you can only choose a numerical column; for the labels you can use a non-numerical column.

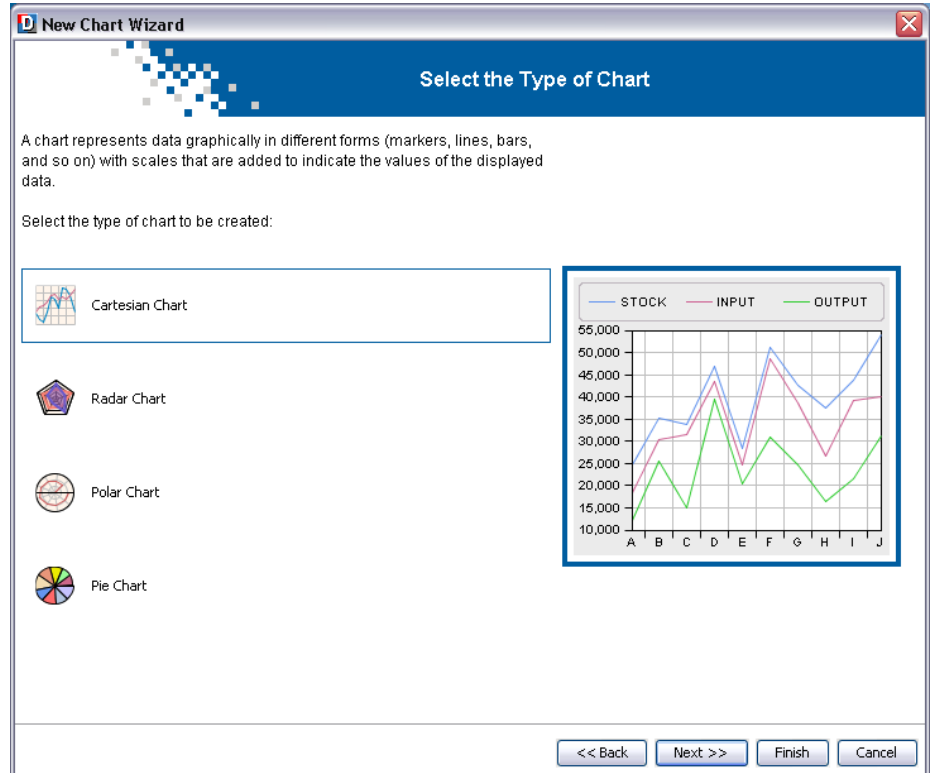
The data mapping is complete when all the columns containing the data that you want to view are mapped to data series.

In this example you can leave the default mapping.

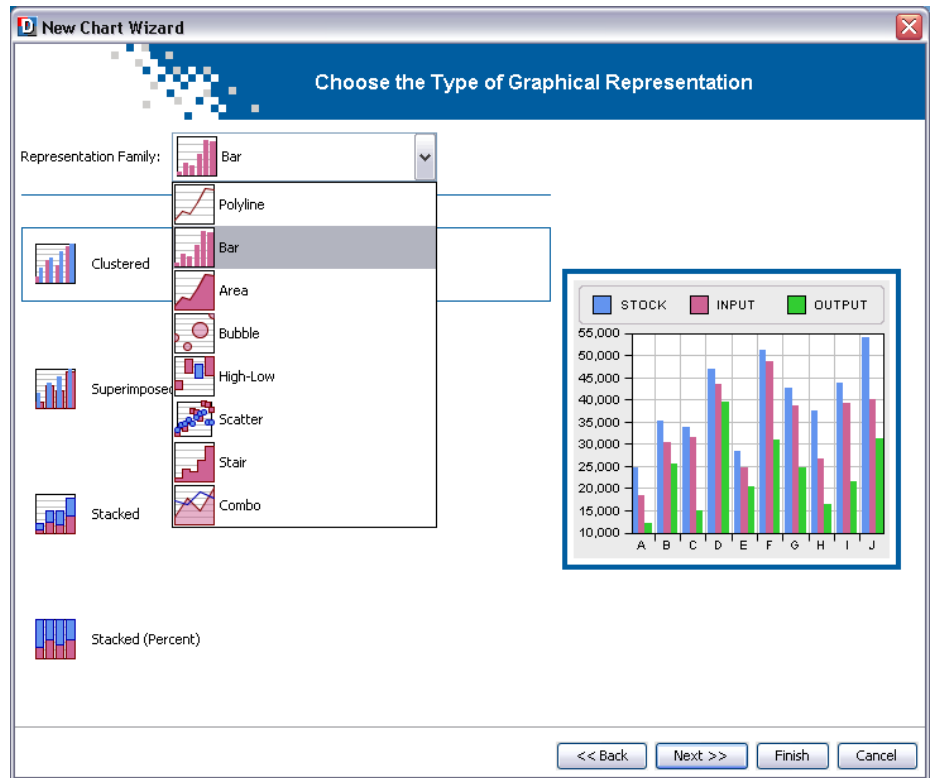
- ◆ Click Next to continue.

Selecting the type of chart

1. Create a Cartesian chart.



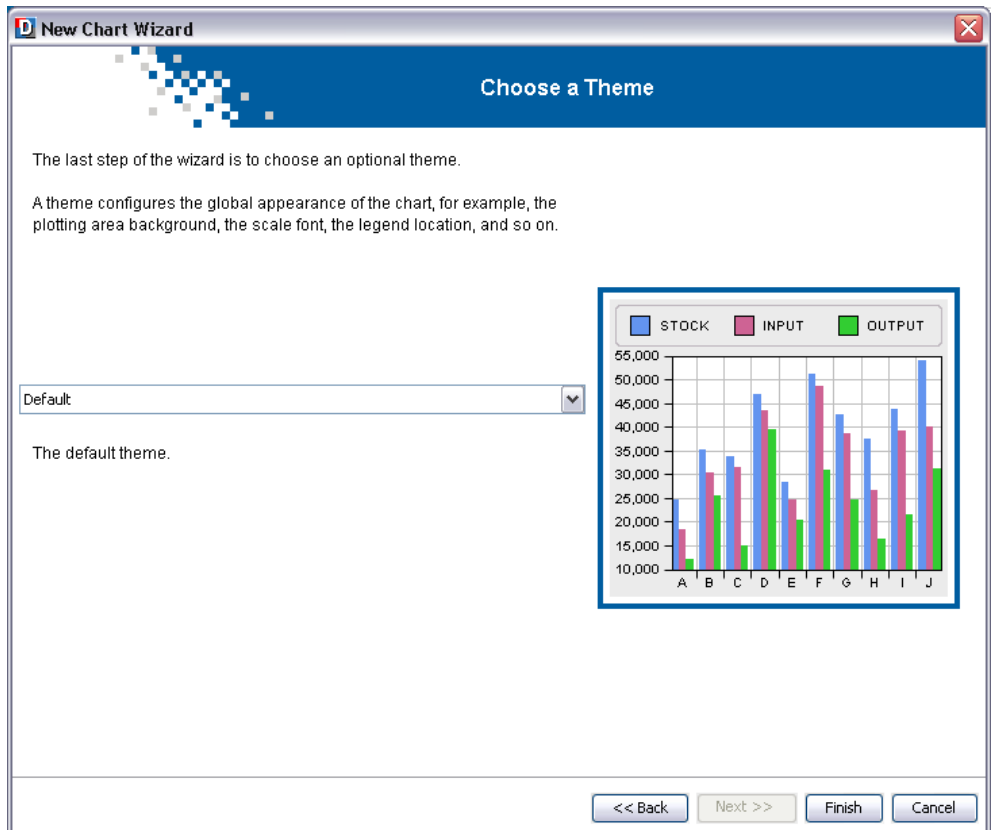
2. Select the option Create a Cartesian chart and click Next.
3. From the Representation Family list, select the option Bar.



4. Select the option Clustered and click Next.

Choosing a theme

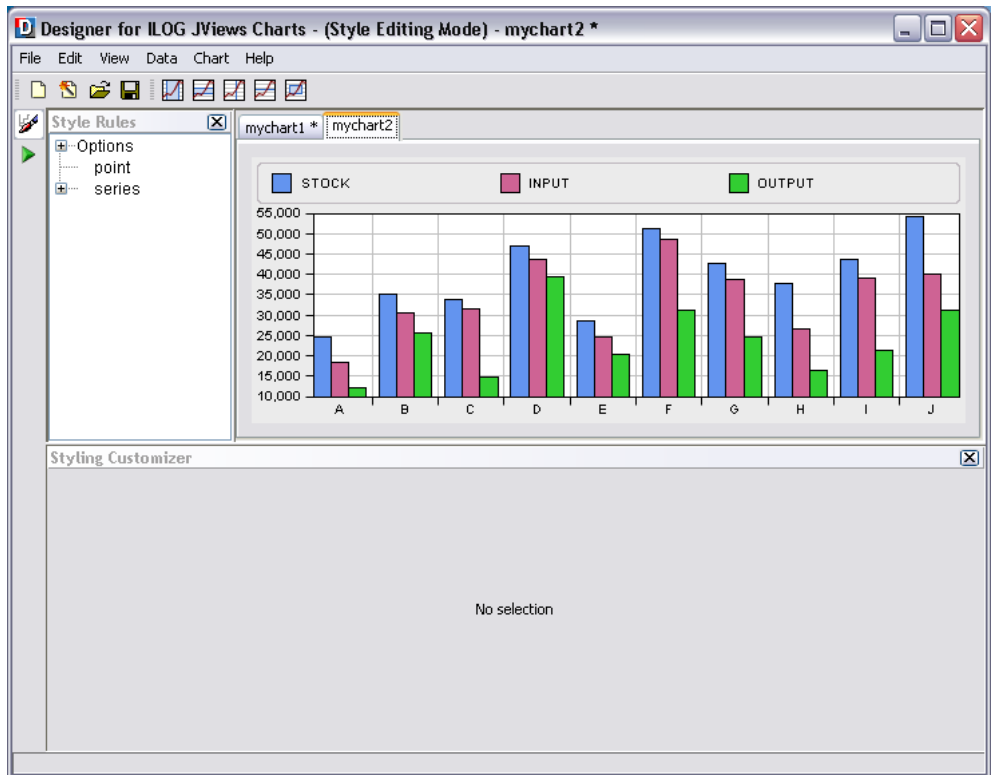
For this example, you can leave the Default theme.



◆ Click Next to end your wizard session.

If you need to make changes, click Back to return to specific wizard pages.

The resulting chart should appear in the Designer, as follows:

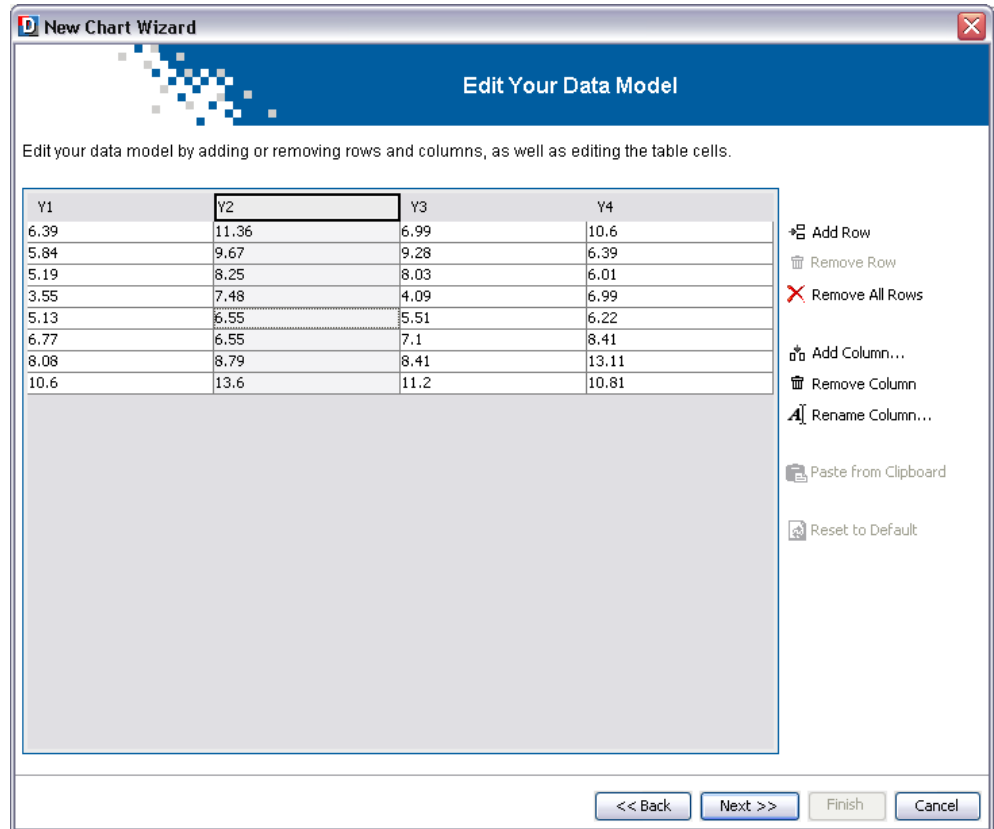


Loading data from In-Memory

Data points are stored in memory with arrays of double primitives. This type of data source supports writing operations such as appending a new data point or changing values of an existing data point. For more information on the in-memory data source, see Populating the chart. in *Introducing JViews Charts*.

Editing your data model

You are going to edit the data model by removing the column “Y2”.



1. Select the column Y2 by clicking one of its cells and then click the Remove Column button.
2. Click Next to continue.

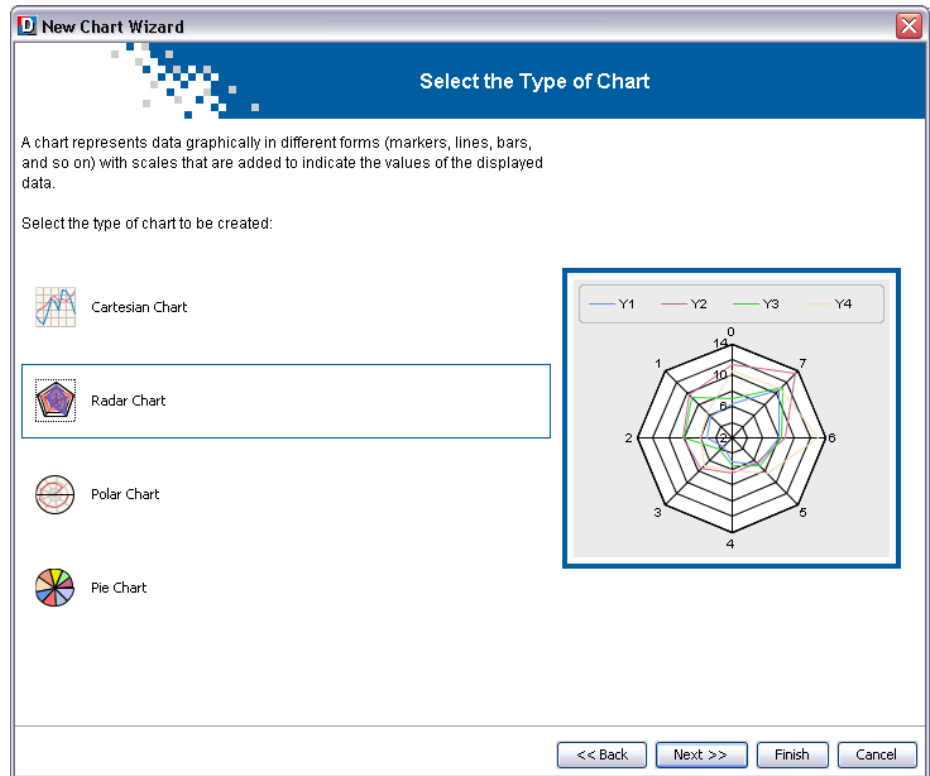
Define the data mapping

In this example, you can leave the default data mapping.

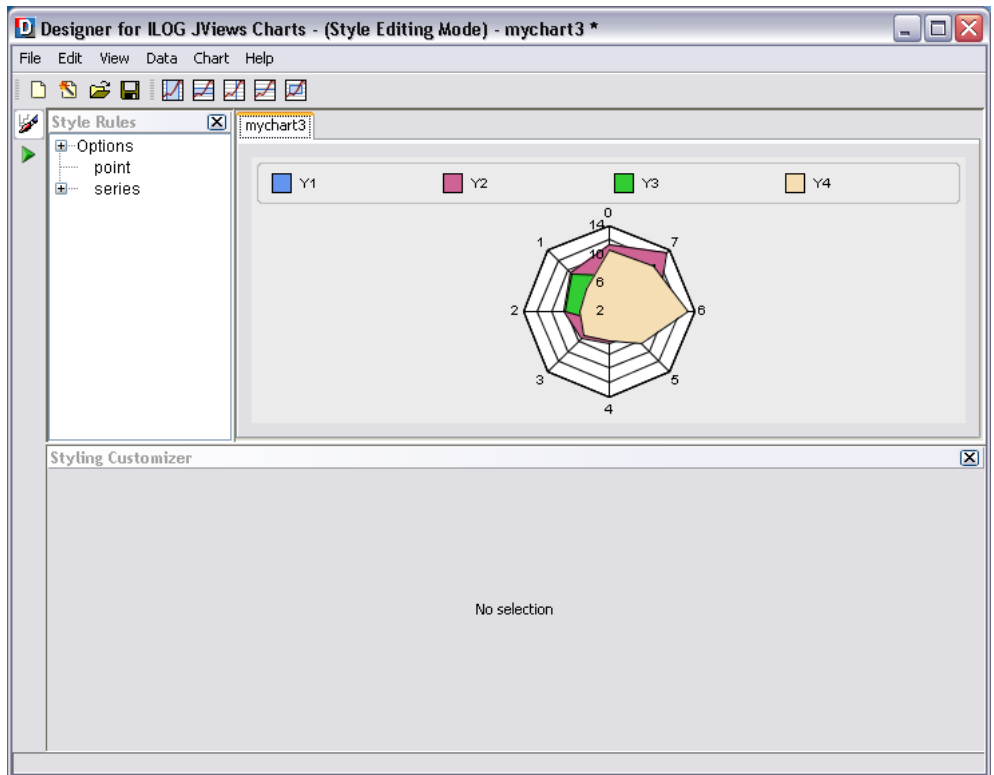
- ◆ Click Next to continue.

Selecting the type of chart

1. Select the option Create a radar chart and click Next to continue.



2. From the Representation Family list select the option Area.



Customizing your chart

Describes how to customize an existing chart.

In this section

Configuring the chart area

Describes how to configure the chart area.

Configuring chart general properties

Describes how to configure the general properties of the chart.

Series graphical representation

Describes the different graphical representation available for the series.

Configuring the legend

Describes how to configure the legend of the chart.

Configuring grids

Explains how to configure the grids.

Configuring scales

Explains how to configure scales.

Configuring 3-D rendering

Explains how to set a 3-D display on your chart.

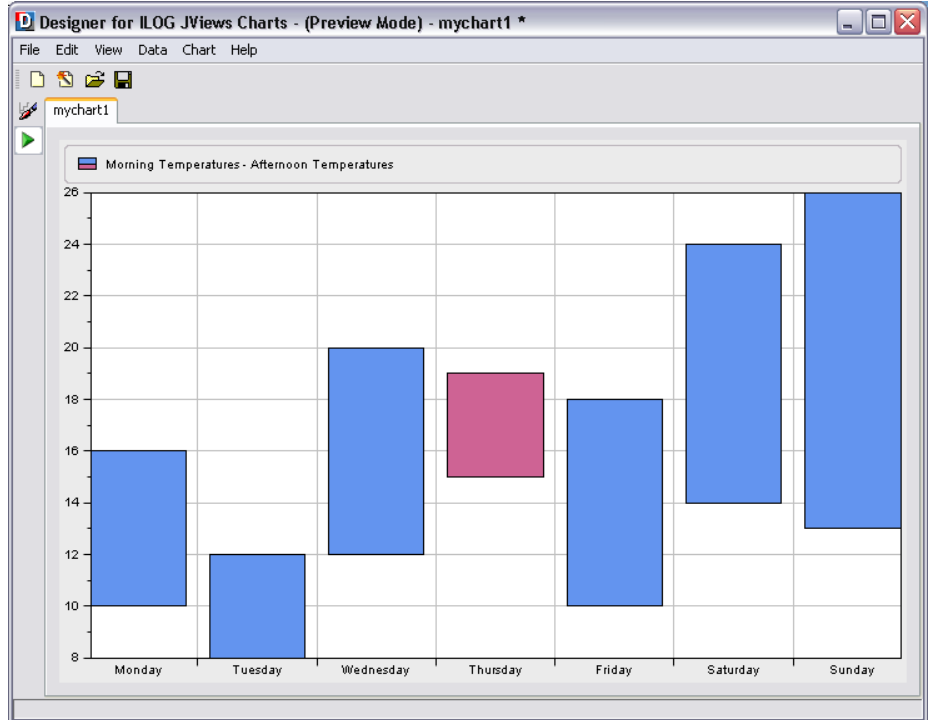
Managing decorations

Describes the various decorations and you can add them to your chart.

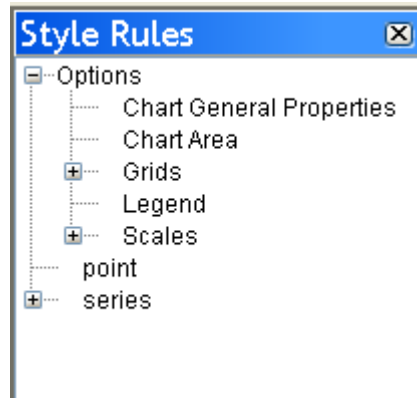
Configuring the chart area

All the examples use the basic Cartesian chart you created in the *Getting Started*.

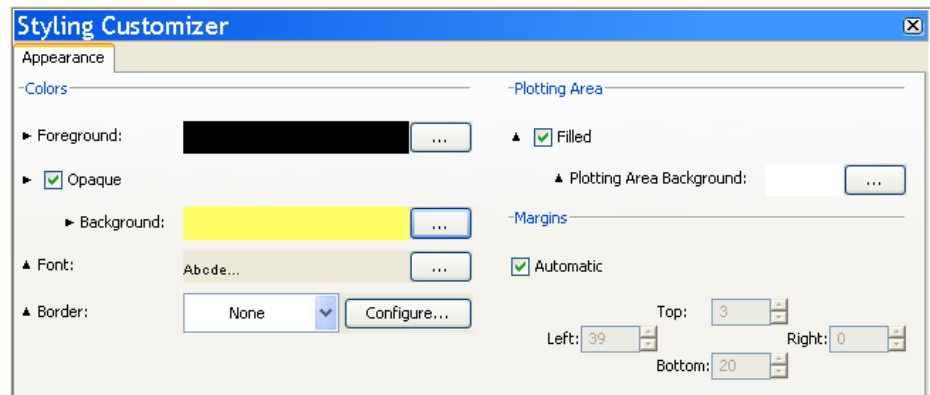
1. Open the project `TemperaturesCartesianChart.icpr` you saved in the *Getting Started*. This is how the basic Cartesian chart appears in the Designer:



2. Select the Chart Area option in the Style Rules tree pane.

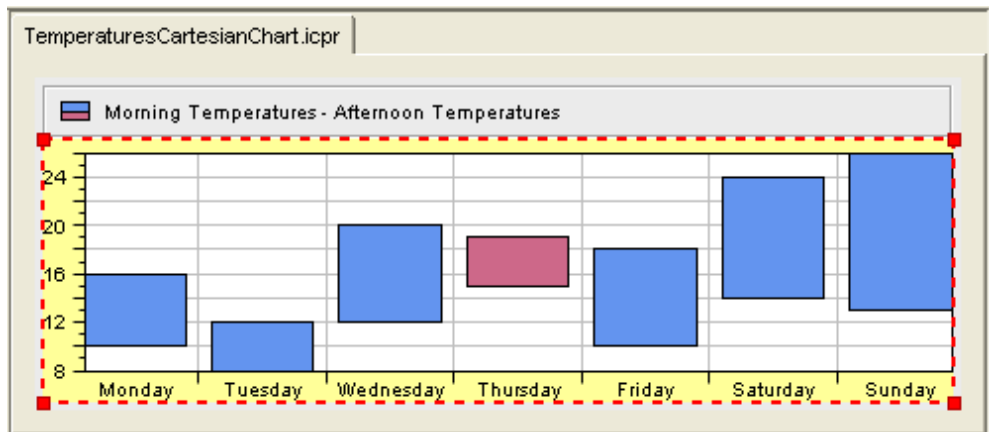


3. Make sure the Styling Customizer pane is active.



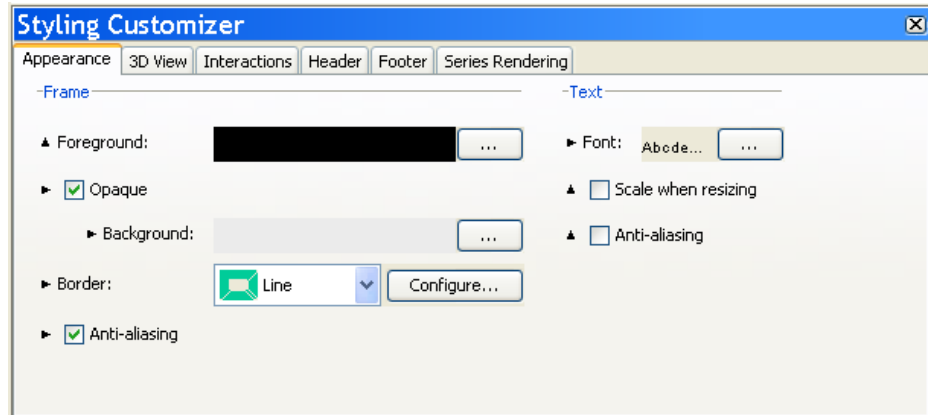
4. Change the background color to yellow and select Opaque.
5. Leave the default settings for the plotting area and margins.

The resulting chart should appear as follows:



Configuring chart general properties

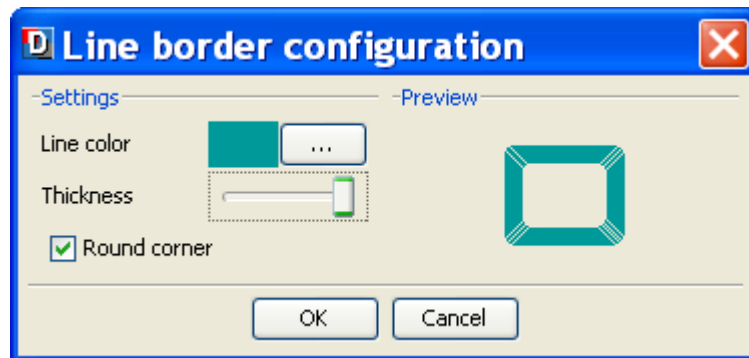
1. Select the Chart General Properties option from the Style Rules tree pane.
2. Make sure the Styling Customizer pane is active.



From the Appearance tab, apply the following configurations:

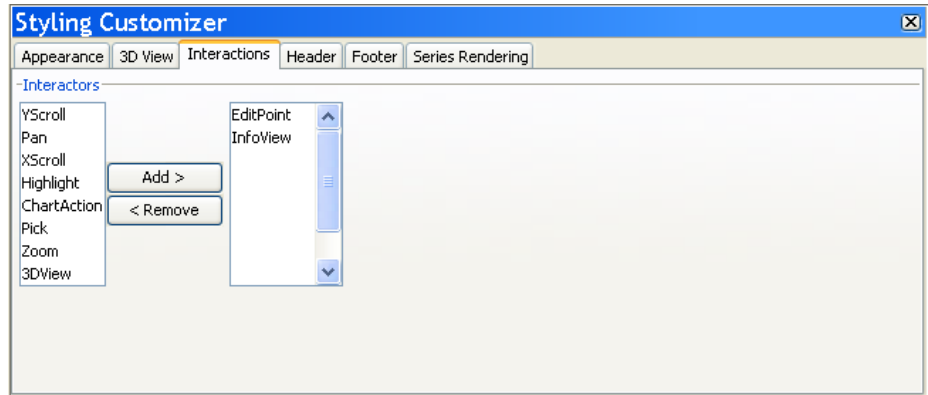
3. Leave the default foreground and background colors.
4. Change the border.

From the border list, choose Line and click the Configure button. Change the line color to green, increase the tickness and check the Round corner option.



From the 3D View tab, you can apply to your chart a three-dimensional display. The type of chart we are treating in this example does not support the 3-D view. To see how to configure the 3-D view, refer to *Configuring 3-D rendering*.

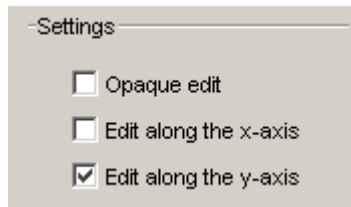
5. From the Interactions tab, apply the following configurations:



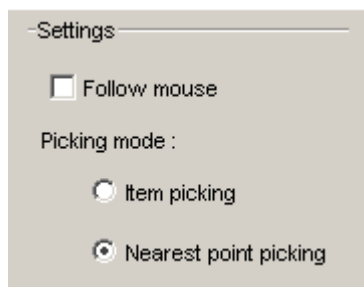
The list on the left displays the available interactors. The list on the right displays the interactors you want to add to your chart.

For the following interactors the JViews Charts Designer allows you to specify additional settings:

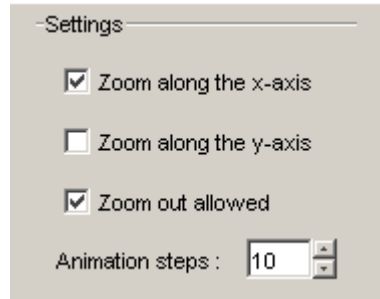
◆ EditPoint Interactor



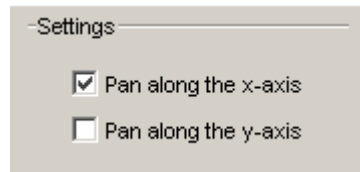
◆ InfoView Interactor



◆ Zoom Interactor



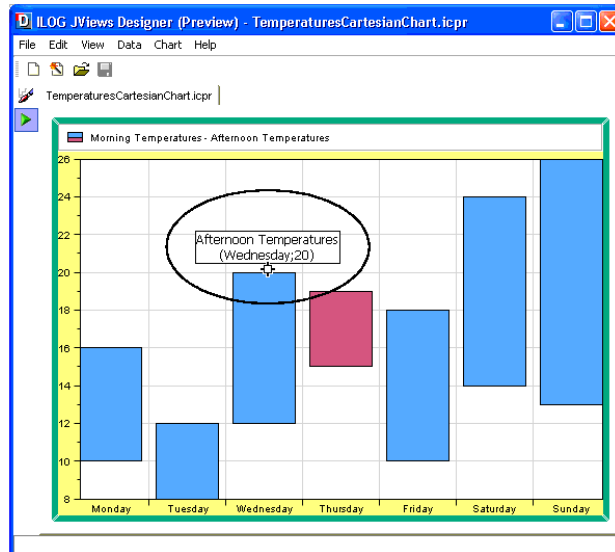
◆ Pan Interactor



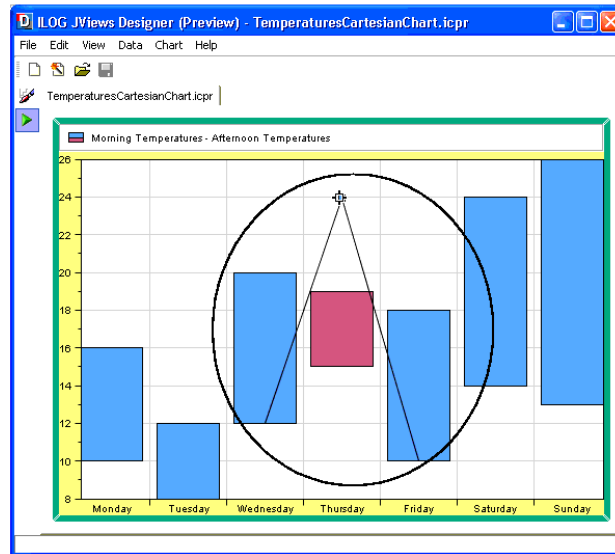
6. Add the `EditPoint` and `InfoView` interactors to your chart. Select them in the Interactors list and click Add.

7. Switch to the Preview mode to make the interactors active.

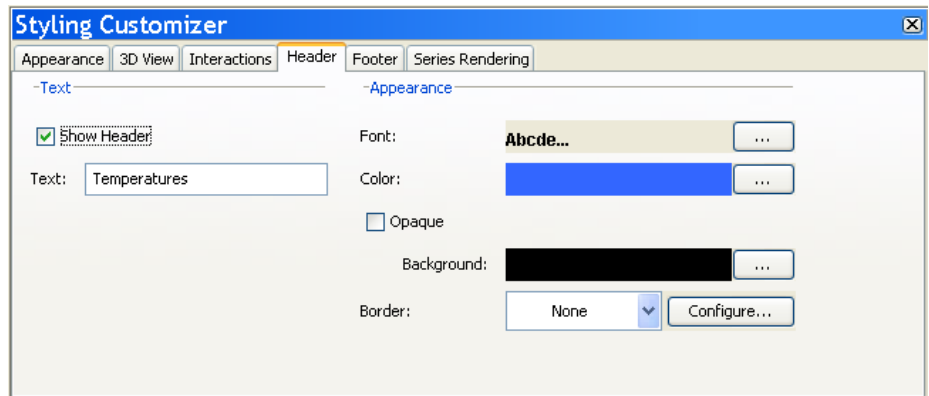
The `InfoView` interactor allows you to display information about a data point whenever you move the mouse over the data point in the data display area.



The `EditPoint` interactor allows you to modify a data point by dragging its graphical representation within the display area.



- Go back to the Style Editing mode and activate the Header tab to add a title at the top of your chart.



- Check the option Show Header to make the title visible.
- In the Text field type the title "Temperatures" and press Enter to validate.
- From the Font editor, change the font properties as indicated below and click Apply:
 - ◆ Font type: Arial
 - ◆ Font size: 12

◆ Bold

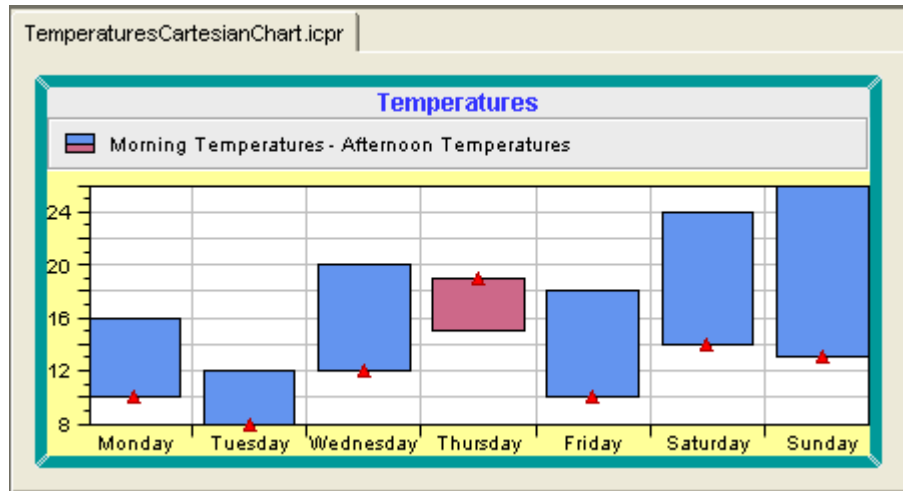
12. From the Color editor, choose the blue color.
13. Leave the default values for the remaining properties.

Note: If you want to add a title at the bottom of your chart, activate the Footer tab and apply the same configurations.

14. On the Series Rendering tab, you can leave the default settings.

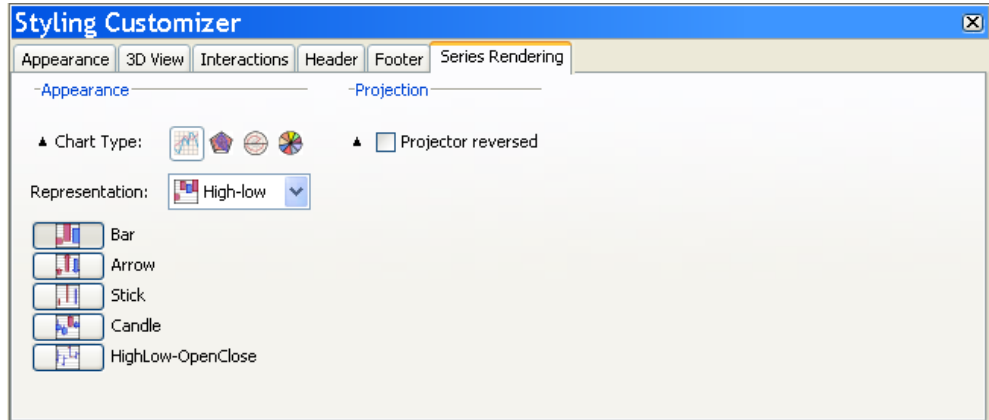
This tab allows you to select a type of chart different from the one you have currently chosen for your chart, or select a different type of representation.

The resulting chart should appear as follows:



Series graphical representation

The Representation drop-down box on the Series Rendering tab lets you choose the type of the graphical representation of the series.



Each representation comes then in different modes. These modes are specific to the selected representation.

POLYLINE Representation

This representation displays the series as polylines. It supports three representation modes:

- ◆ **Superimposed Mode**
Polylines are drawn on top of each other.
- ◆ **Stacked Mode**
Polylines are stacked, so that each one displays the contribution of a y-value in a set of several y-values.
- ◆ **Stacked (Percent) Mode**
The contribution of a y-value is computed as a percentage of all the y-values for a given x-value.

BAR Representation

This representation displays the series as vertical or horizontal bars (orientation is determined from the chart projector configuration). It supports four representation modes:

- ◆ **Clustered Mode**
Bars are laid out in clusters, each cluster representing the set of y-values corresponding to a given x-value.

◆ Superimposed Mode

Bars are drawn on top of each other.

◆ Stacked Mode

Bars are stacked, so that each one displays the contribution of a y-value in a set of several y-values.

◆ Stacked (Percent) Mode

The contribution of a y-value is computed as a percentage of all the y-values for a given x-value.

◆ Stacked (Diverging) Mode

Bars are stacked. Positive and negative values are displayed separately: positive values in bars with an upward direction, negative values in bars in the opposite direction.

AREA Representation

This representation displays the series as areas. It supports three representation modes:

◆ Superimposed Mode

Areas are drawn on top of each other.

◆ Stacked Mode

Areas are stacked, so that each one displays the contribution of a y-value in a set of several y-values.

◆ Stacked (Percent) Mode

The contribution of a y-value is computed as a percentage of all the y-values for a given x-value.

BUBBLE Representation

This representation displays a two-dimensional data model as bubbles of variable size. The data model should be described by two data sets, the first data set determining the location of the bubbles, and the second data set determining the size of the bubbles. There is one representation mode.

HILO Representation

This representation displays two data sets with High/Low items. It supports five representation modes.

◆ Bar Mode

This mode creates a renderer for every pair of data sets contained in the data source and display them as bars.

◆ Arrow Mode

This mode creates a renderer for every pair of data sets contained in the data source and display them as arrows.

◆ Stick Mode

This mode creates a renderer for every pair of data sets contained in the data source and display them as sticks.

◆ Candle Mode

This mode handles two pairs of data sets: the first pair for the low/high values, the second pair for the open/close values. The low/high values are displayed as a stick, while the open/close values are displayed as a bar.

◆ HighLow-OpenClose Mode

This mode handles two pairs of data sets as for the OpenClose mode. The mode differs from the previous mode by its representation. The low/high values are rendered with horizontal marks, the Open/Close values with as a bar.

SCATTER Representation

This representation displays a data set with scattered graphical markers. It supports one mode.

STAIR Representation

This representation represents a transition between two values as a stair.

◆ Superimposed Mode

Stairs are drawn on top of each other.

◆ Stacked Mode

Stairs are stacked, so that each one displays the contribution of a y-value in a set of several y-values.

◆ Stacked100 Mode

The contribution of a y-value is computed as a percentage of all the y-values for a given x-value.

COMBINED Representation









This representation displays series with multiple representations. Contrary to the previous representations, it allows you to render each data set by a different representation, mixing a subset of the representation mode listed above. For example, it can render a data set with a scatter representation and the other data sets with a stacked bar representation.

Note: The drawing order of the graphical representation is determined by the series order in the data source, the first series being drawn first (that is, "below").

By checking the option Projector reversed you can reverse the projector.

A reversed projector swaps the meaning of the abscissa and ordinate coordinates of a point. For example, a reversed Cartesian projector projects x-data values along the y-axis of the screen.

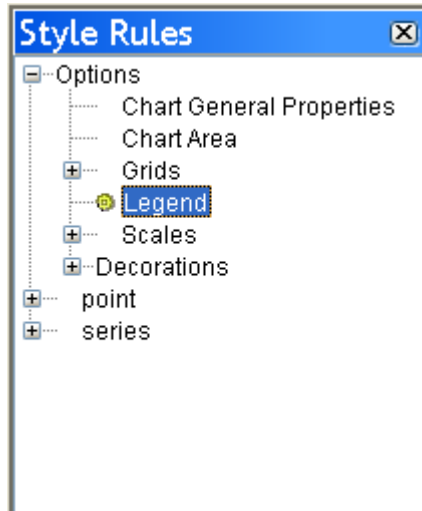
Cartesian Orientation illustrates the different Cartesian orientations that can be specified by using this property in conjunction with the reversed property of an axis:

Projector Reversed	X-Axis Reversed	Y-Axis Reversed	Cartesian Orientation
false	false	false	
false	false	true	
false	true	false	
false	true	true	
true	false	false	
true	false	true	
true	true	false	
true	true	true	

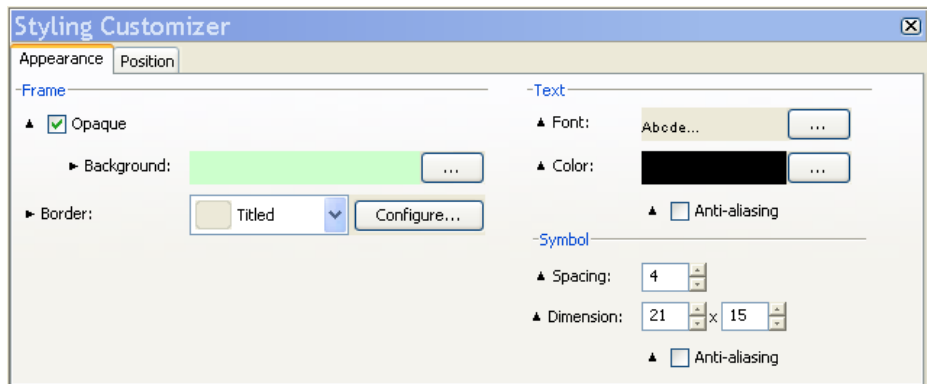
Cartesian Orientation

Configuring the legend

1. Select the Legend option from the Style Rules tree pane.



2. Make sure the Styling Customizer pane is active. From the Appearance tab, apply the following configurations:



3. Change the background color to light green.
4. Check the option Opaque.
5. Leave the default type of border, which is Titled.

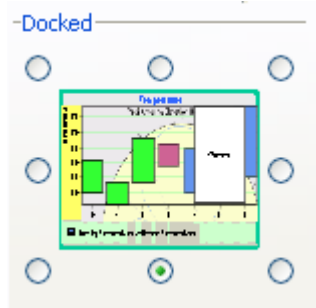
If your legend has a title, this would be visible only if you select the type of border Titled.

6. Add a title to the legend.

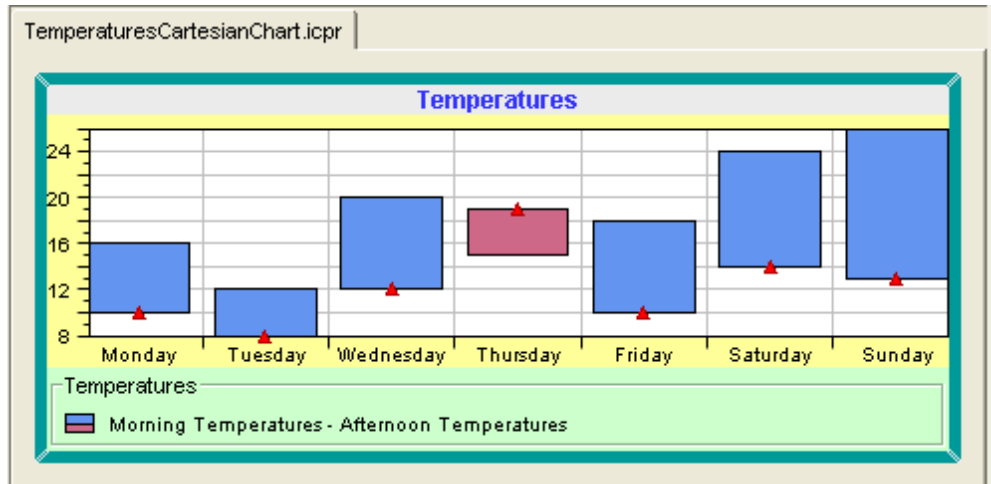
Click the Configure button, type "Temperatures" in the Title field and click OK.

7. Leave the default settings for the remaining properties.
8. Change the position of the legend on the chart.

Activate the Position tab and from the Docked options, select the bottom of the chart.

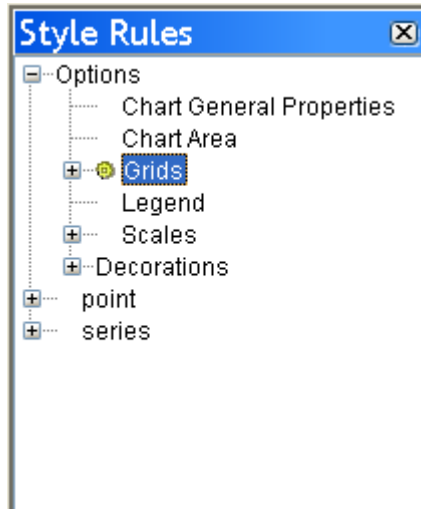


The resulting chart should appear as follows:

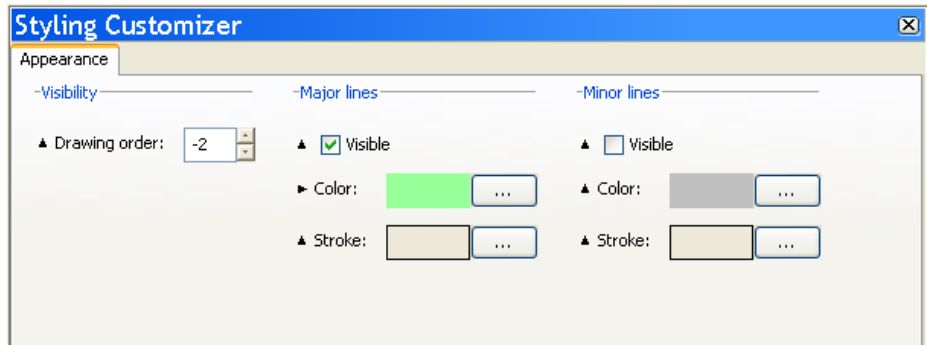


Configuring grids

1. Select the Grids option from the Style Rules tree pane.



2. Make sure the Styling Customizer pane is active.



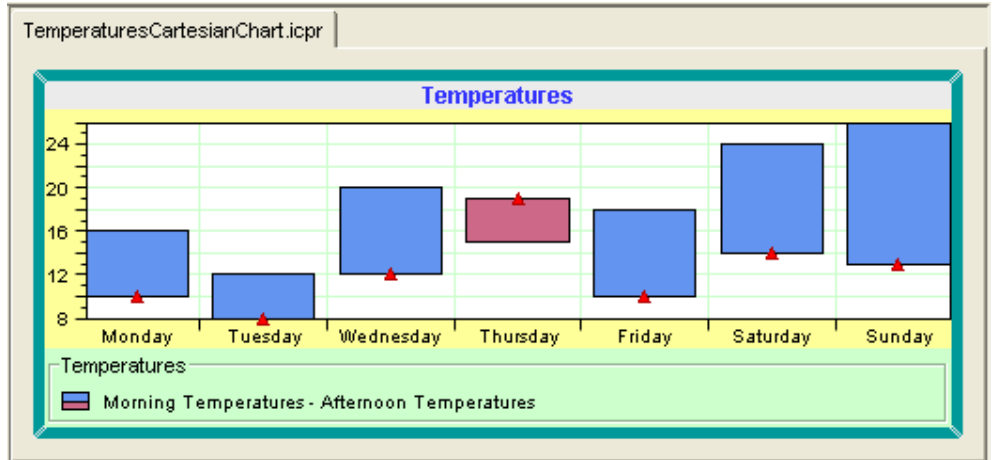
With the Y grid selected, proceed as follows:

3. Set the drawing order to -1.

Note: The drawing order lets you control the position of a decoration in the drawing queue of a chart. Decorations with a negative drawing order are drawn below the chart representations, while decorations with a zero or positive drawing order are drawn above the chart representations.

4. Set the major lines as visible.
5. Change the color to light green.
6. Leave the default settings for the minor lines.
With the X grid selected, proceed as follows:
7. Set the drawing order to -2.
8. Apply the same configurations you applied to the Y grid.

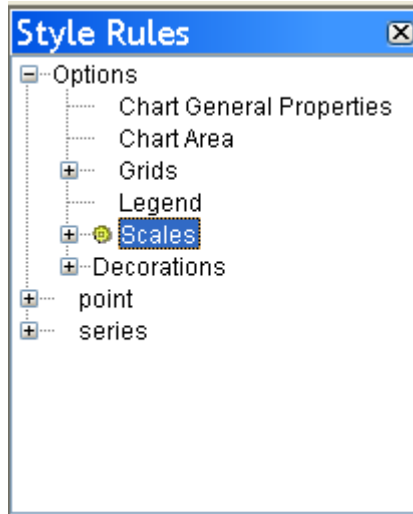
The resulting chart should appear as follows:



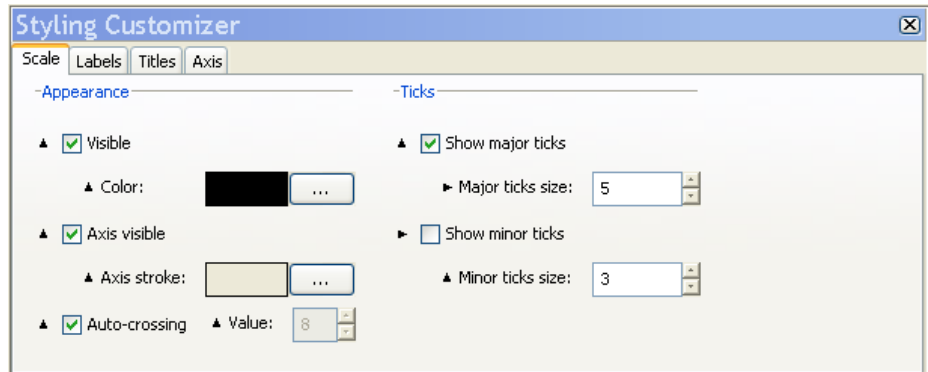
Note: To write your own grid see Writing a new grid in *Developing with the SDK*.

Configuring scales

1. Select the Scales option from the Style Rules tree pane.



2. Make sure the Styling Customizer pane is active.



From the Scale tab, apply the following configurations:

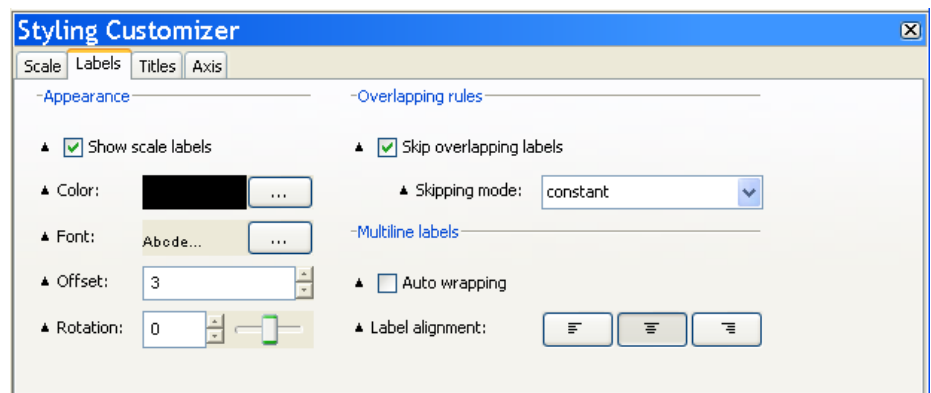
3. Set the scales visible and leave the default color.

Note: Showing or hiding a scale affects the drawing area bounds, therefore a layout is automatically performed on the chart area when you check the Visible option.

4. Set the axis visible and leave the default axis stroke.
5. Leave the default position of the scales on the chart, which is Auto-crossing.

Note: The position of a scale is defined with respect to the scale dual axis according to a given data value. This data value is the value on the dual axis where the scale axis crosses it, and is called the scale crossing value.

6. Show major ticks and set their size to 5.
7. Do not show minor ticks.
8. Activate the Labels tab and apply the following configurations:



9. Show the scale labels.
10. You can rotate the scales vertically or horizontally by either entering a value in the Rotation field or moving the slider.

In Cartesian charts, the abscissa and ordinate scales can be oriented either horizontally or vertically. In Polar charts, the circular x-axis can be oriented clockwise or counterclockwise and the radial ordinate axis can be oriented at any angle.

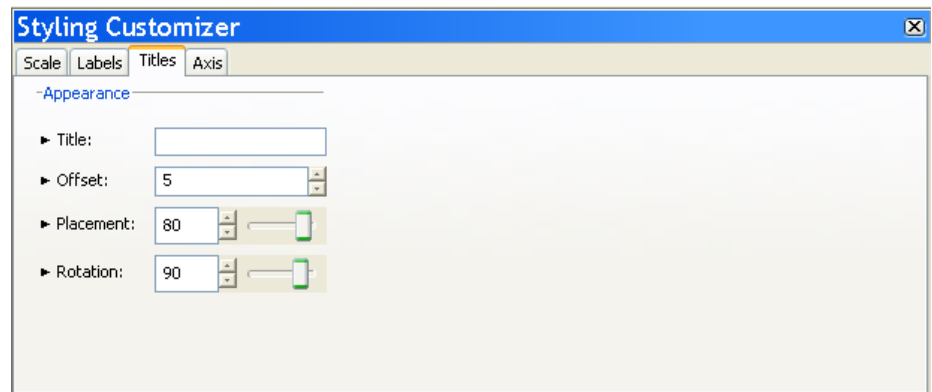
11. Check the option Skip overlapping labels.

Depending on the values and the way labels are formatted, it may occur that steps labels overlap each other because of the text length.
12. Leave the Skipping mode set to CONSTANT.
 - ◆ CONSTANT mode: the scale computes a constant number of labels to skip to avoid overlapping.
 - ◆ ADAPTIVE mode: scale considers every label from the first to the last, and determines which one should be skipped to avoid overlapping.
13. In case of multiline labels, check the option Autowrapping.

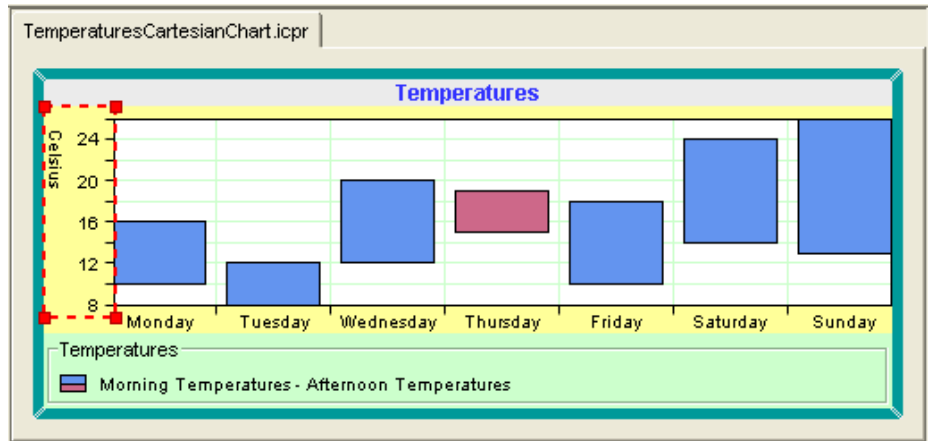
This option toggles the automatic wrapping of step labels. When the wrapping is turned on, the scale tries to wrap the labels into several lines according to the space available between two steps. The wrapping also takes into account the explicit new line characters.

Note: The label wrapping does not check the orientation nor the type of the scale. Similarly, it does not take into account the rotation of scale labels. It should only be enabled for scales whose labels are parallel to the axis.

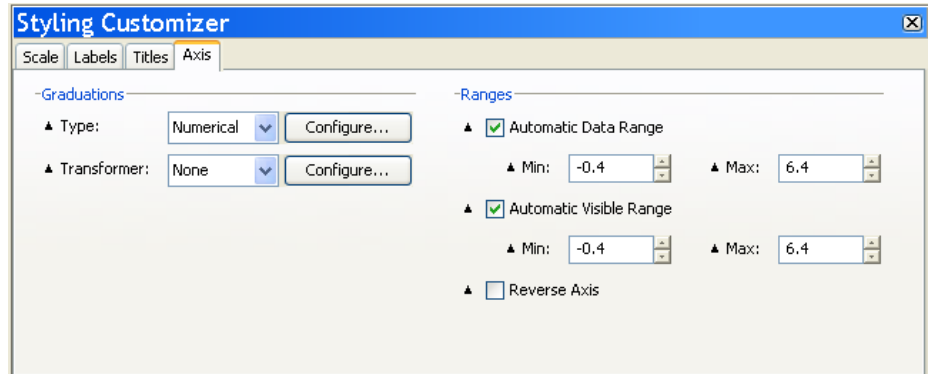
14. Activate the Titles tab and apply the following configurations:



15. Select the Y scale and add the title "Celsius".
Type "Celsius" in the Title text field and press Enter to validate.
16. Specify an offset value of 5.
17. Set the title at the top of the scale.
You can either set the Placement value to 80 or move the slider.
18. Place the title in the vertical direction.
You can either set the Rotation angle to 90 degrees or move the slider.
The resulting chart should appear as follows:



19. Activate the Axis tab and apply the following configurations:

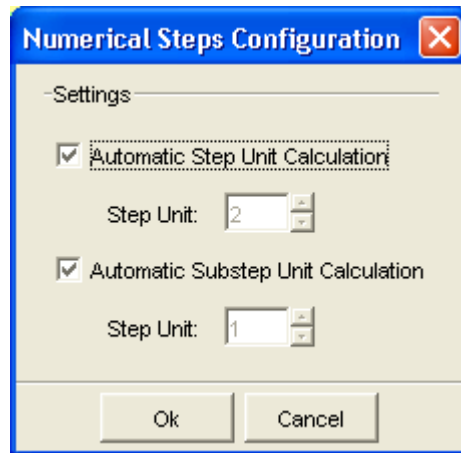


20. Set the graduation type to numerical.

Four graduation types are available:

- ◆ numerical: for scales that display standard numerical values.
- ◆ time: for scales that display time values.
- ◆ category: for scales that display categories.
- ◆ logarithmic: for logarithmic scales.

21. You can click the Configure button to configure the numerical steps graduation. In this example you are going to leave the default settings.



The step and substep unit values are automatically computed at run time. You can disable the automatic steps calculation by defining a specific step unit.

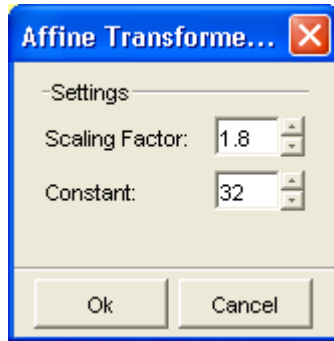
22. You are now going to use the Transformer option to express the temperatures in Fahrenheit.

Note:

The Transformer option allows you to apply a transformation to the coordinates along a given axis. The available types of transformer are:

- ◆ Affine: the transformation is defined by scaling and constant coefficients.
- ◆ Logarithmic: the transformation is defined by a logarithmic base.
- ◆ Local Zoom: the transformation is defined by a scaling factor to data values within a given range.

23. Select the Y axis.
24. Set the Transformer option to Affine.
25. Click the Configure button and enter the following values:
 - ◆ Scaling factor: 1.8
 - ◆ Constant: 32Click OK to validate your choice.



The temperatures are now expressed in Farenheit and you can see how their values have changed on the axis (they range from 50 to 75).

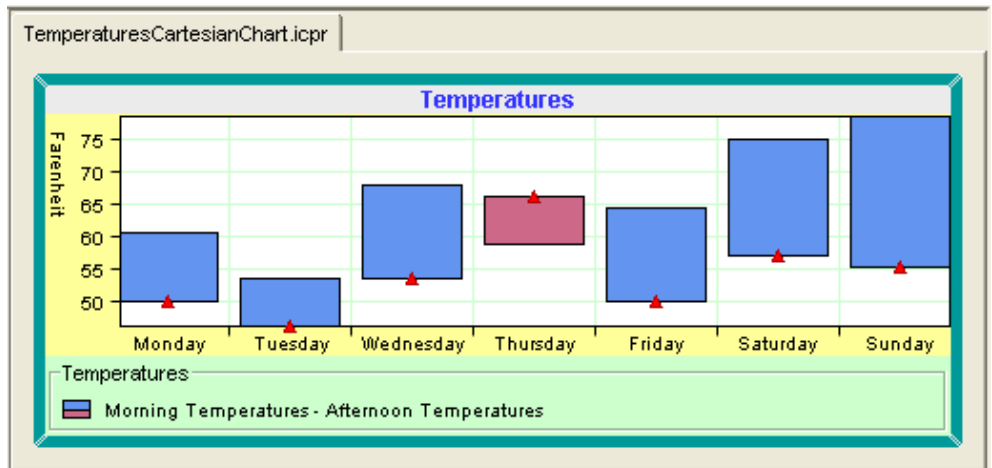
To be consistent with the new type of temperature, you need to change the title as well.

26. Go back to the Titles tab and type "Farenheit" in the Title field. Press Enter to validate.
27. Leave the default settings for Data Range and Visible Range.

- ◆ Data Range: defines the limits of the data values along a specified axis.
- ◆ Visible Range: defines the visible data interval along a specified axis.

If you set these options to Automatic mode, the ranges are automatically computed to fit the data actually displayed by the chart.

The resulting chart should appear as follows:



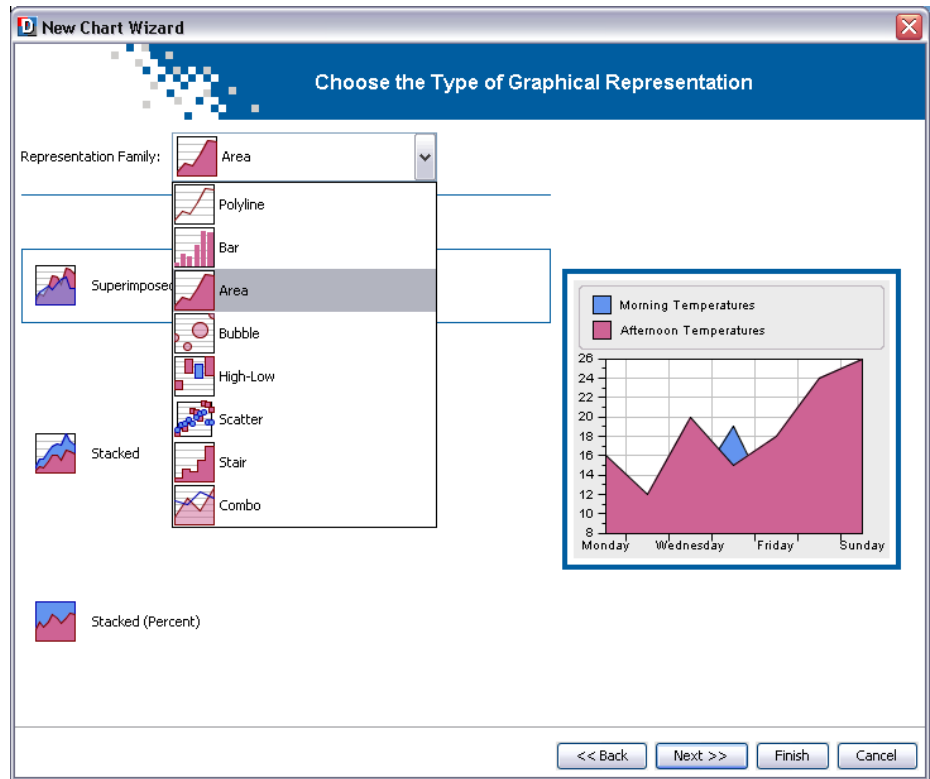
Configuring 3-D rendering

The Designer allows you to switch from a two-dimensional to a three-dimensional display. Only Cartesian and pie charts support 3-D rendering.

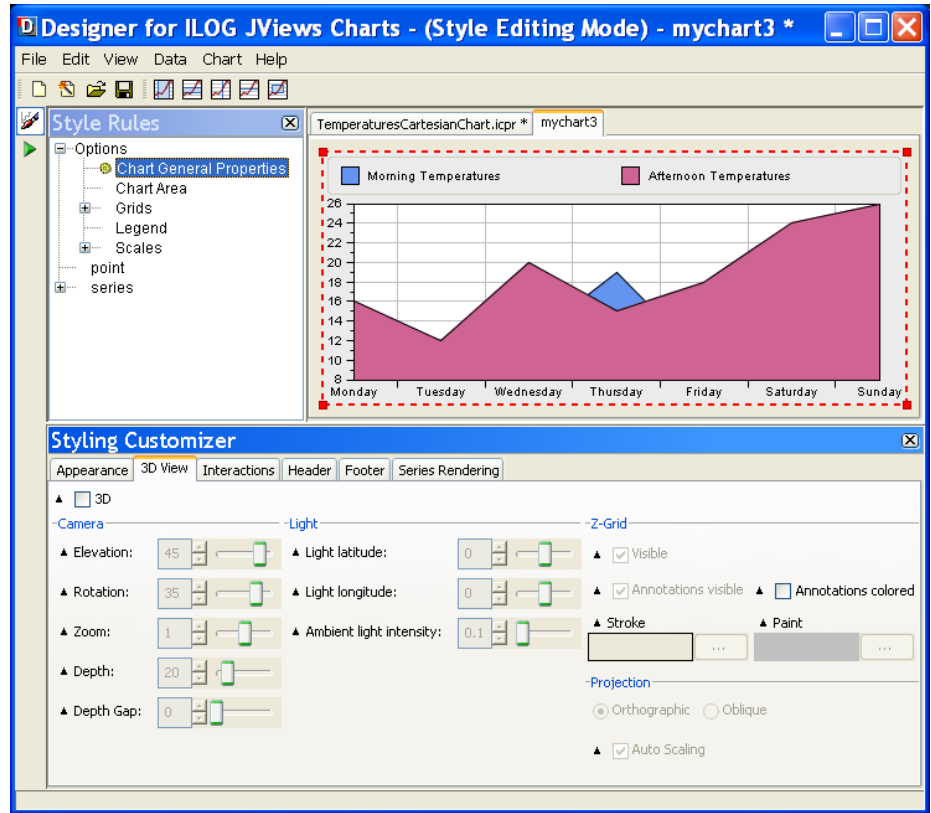
To better appreciate the 3-D rendering benefits, you are going to create a basic Cartesian chart and choose Area as graphical representation.

To create the basic Cartesian chart, see *Creating a new basic chart* and follow the same steps until you reach the step Choose the Type of Graphical Representation. Once there, proceed as explained below.

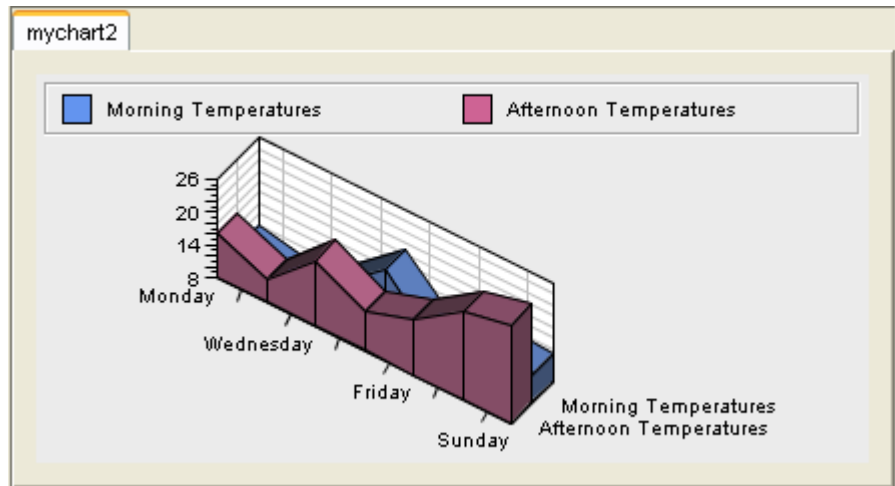
1. From the Representation Family list, select Area.



2. Choose Superimposed and click the Next button.
3. Leave the default Theme and click the Next Button.
4. Click Finish to exit the Wizard.



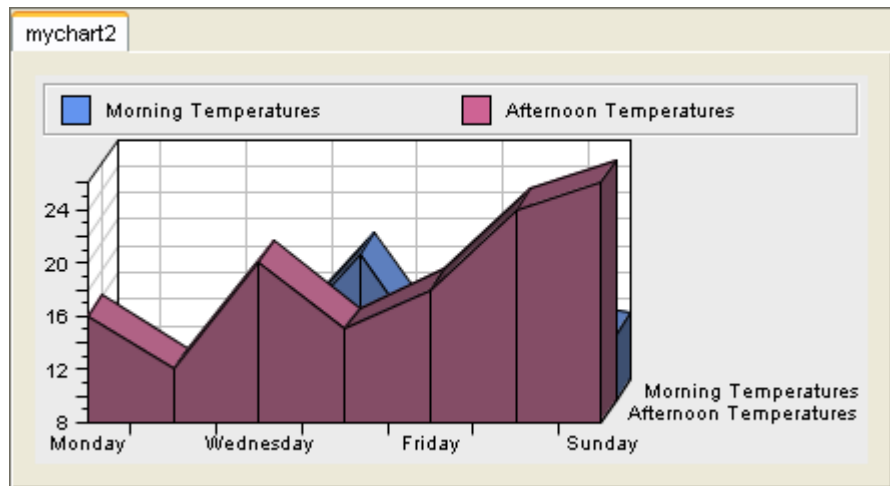
5. Select Chart General Properties from the Style Rule tree pane and make sure the 3-D View tab is active in the Styling Customizer.
 6. Check the 3-D option.
- Your chart will be displayed with the 3-D view.



7. Select the Oblique projection.

8. Check the option Auto-scaling.

The resulting chart should appear as follows:



Managing decorations

The Designer allows you to add the following decorations to your chart:

- ◆ grids
- ◆ data indicators
- ◆ image

Decorations are drawn according to a *drawing order*. The drawing order lets you control the position of a given decoration in the drawing queue of a chart. This drawing order is used to define the position of the decoration relative to:

- ◆ Other chart decorations.
- ◆ Graphical representations of the chart data.

A chart handles decorations as an ordered list according to the decorations drawing order: decorations with the lowest drawing order are drawn first, decorations with the highest drawing order are drawn last.


The drawing order also defines whether a decoration should be drawn above or below the graphical representations of the chart data: decorations with a negative drawing order are drawn below the chart representations, while decorations with a zero or positive drawing order are drawn above the chart representations.

To add and configure grids for your chart:

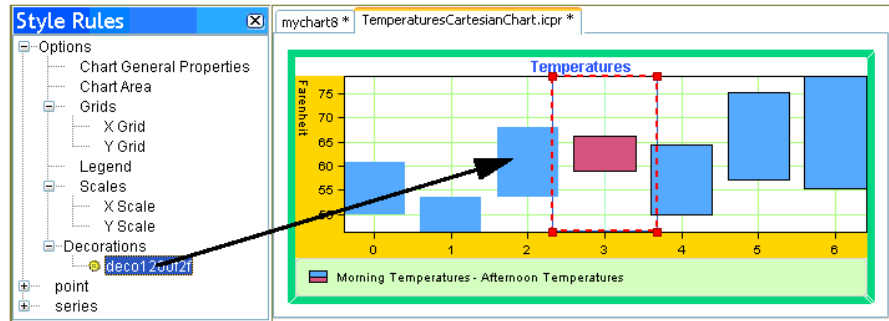
- ◆ See *Configuring grids*. More information on grids can also be found in the section *Grids in Introducing JViews Charts*.

You are going to decorate your chart by highlighting the week-end period using a range indicator which represents a data interval equal to the week-end period.

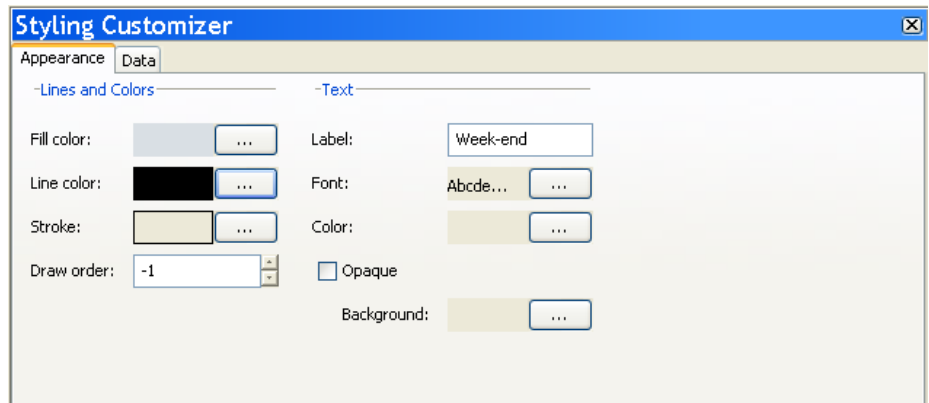
To add data indicators to your chart:

1. With your `TemperaturesCartesianChart.icpr` project open, proceed as follows:
2. Select the X scale in the Style Rules tree pane.
3. Click the  icon on the toolbar to add a range indicator along the X axis.

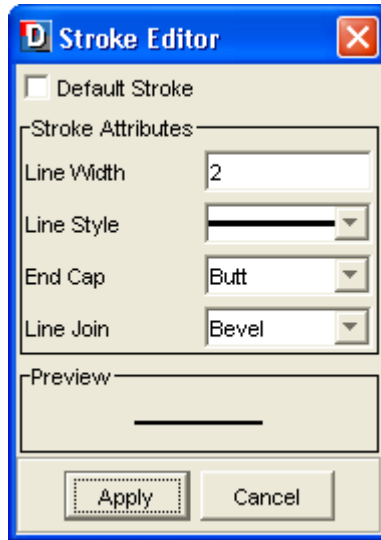
You will notice that a range indicator is visible on the chart area and a new decoration is added to the Style Rules pane under Options:



4. Select the new decoration and activate the Appearance tab on the Styling Customizer pane.



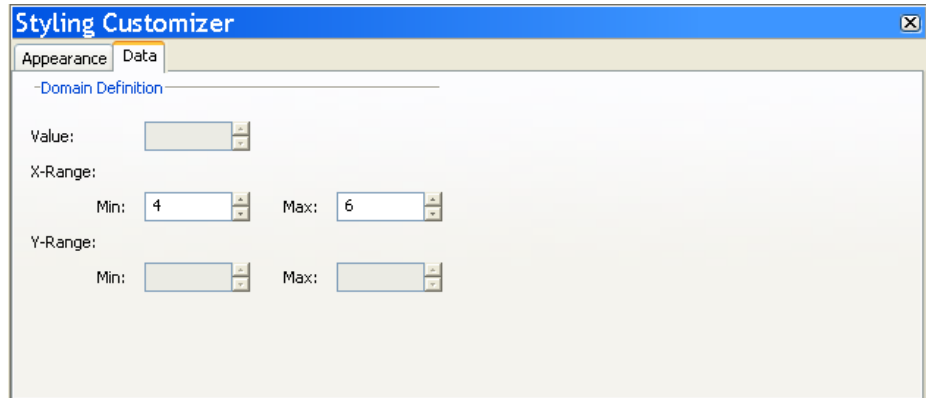
5. In the Fill Color option, choose a transparent color.
6. Change the Line color to black.
7. Modify the Stroke. From the Stroke Editor, change the Line Width to 2.



8. Set the drawing order to 0.

Note: The drawing order lets you control the position of a decoration in the drawing queue of a chart. Decorations with a negative drawing order are drawn below the chart representations, while decorations with a zero or positive drawing order are drawn above the chart representations.

9. Add a label.
Type “Week-end” in the label text field and press Enter to validate.
10. Leave the default settings for the remaining properties.
11. Activate the Data tab and apply the following configurations:

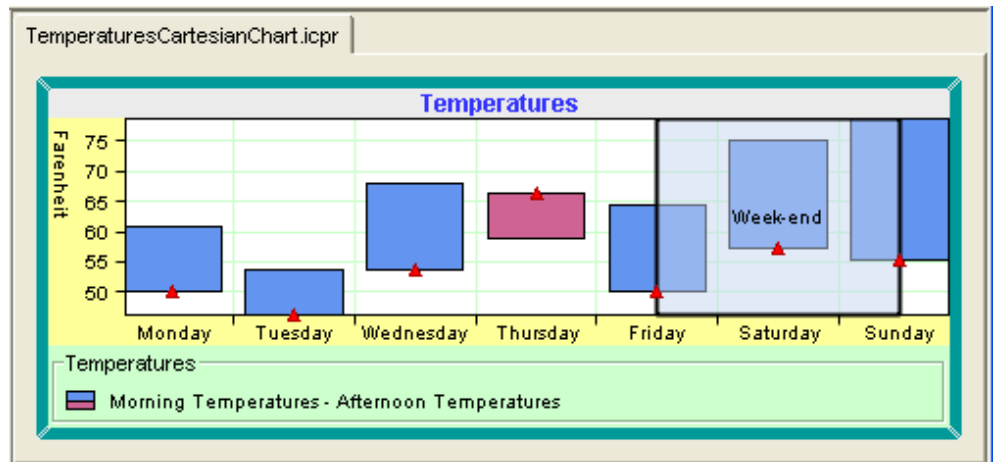


12. On the X-Range option, set the Min value to 4 and the Max value to 6.

The settings on the Data tab correspond to the type of indicator you have chosen to apply:

- ◆ Value indicator: specify a value for the Value option.
- ◆ Range indicator: specify a min/max value for the X-Range or Y-Range.
- ◆ Window indicator: specify a min/max value for both the X-Range and Y-Range.

The resulting chart should appear as follows:

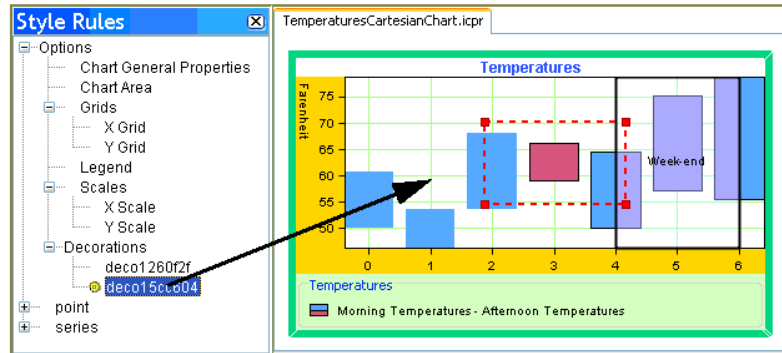


To add an image to your chart:

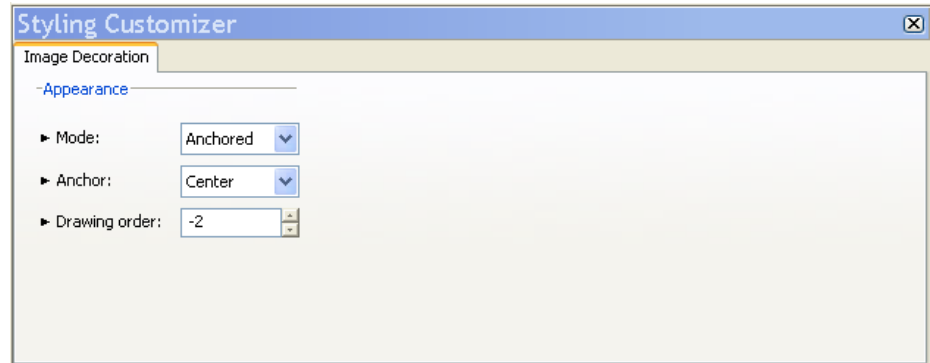
1. From the menu bar, choose *Chart>Image Decoration*.
2. Browse to the image you want to add and click the Open button.

Note: The following formats are supported: GIF, JPEG, PNG.

You will notice that the image appears on the background of the chart area and a new decoration is added to the Style Rules pane under Options:

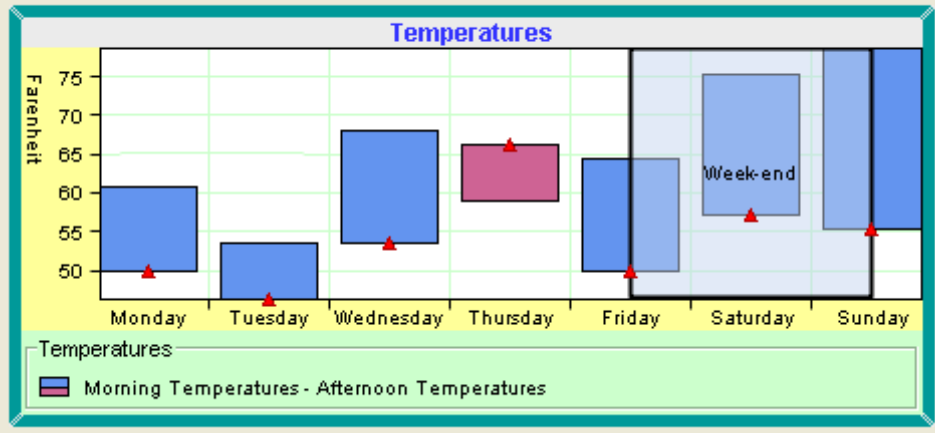


3. Activate the Styling Customizer and apply the following configurations:



4. Set the option Mode to ANCHORED to locate the image at a fixed location.
5. Set the option ANCHOR to NORTH-WEST to locate the image on the upper-left corner of your chart.
6. Set the drawing order to -2.

The resulting chart should appear as follows:



Managing data style rules

Explains how to use the Style Rule Wizard to specify rendering attributes for data series and single data points.

In this section

The Create Style Rule wizard

Explains how to customize the global appearance of your chart by creating or changing the styles rules that define the styling of the data.

Creating rules on data

Explains how to create rules on point or series.

Data points

Explains how to create rules on data points.

Data series

Explains how to create rules on data points.

The Create Style Rule wizard

You can customize the global appearance of your chart by creating or changing the styles rules that define the styling of the data.

- ◆ To open the Create Style Rule Wizard, choose *Edit>Create Style Rule* or right-click a rule and choose Create Style Rule in the pop-up menu.

The Create Style Rule Wizard simplifies the definition of data rules which can include attribute conditions.

Creating rules on data

Through the Create Style Rule Wizard you can create rules on the following object types: point or series. If you have defined your own classes in the data model, you may also have user-defined types. The plus and minus signs allow you to add or remove attribute conditions to which the rule applies.

Attribute conditions

An attribute condition tests one or more attributes against a value. If you select an attribute, you must then set a condition on its value by selecting an operator from the operator menu and entering a value. You can set several attribute conditions.

The operator menu reflects the type of the attribute: x, y, index, label, seriesIndex and seriesName.

For label and seriesName attributes, the operator menu offers:

- ◆ exists and not null
- ◆ equals
- ◆ does not equal
- ◆ contains

For x, y, index and seriesIndex attributes, the operator menu offers:

- ◆ exists and not null
- ◆ equals
- ◆ does not equal
- ◆ is greater than
- ◆ is less than
- ◆ is greater than or equal to
- ◆ is less than or equal to

Data points

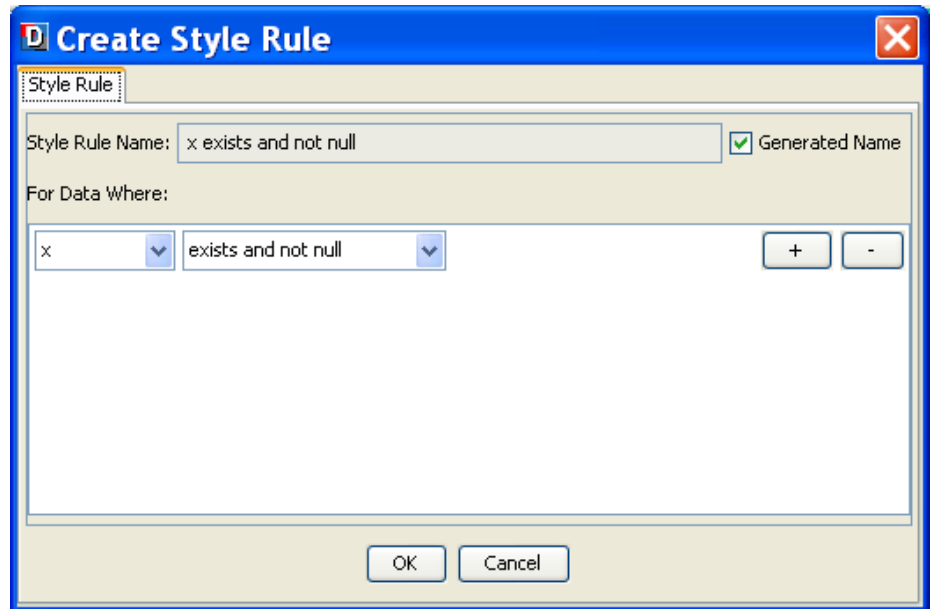
On data points, you can define the following attributes:

- ◆ x: The x-value of the data point.
- ◆ y: The y-value of the data point.
- ◆ index: The index of the data point in the data set.
- ◆ label: The label of the data point.
- ◆ seriesIndex
- ◆ seriesName

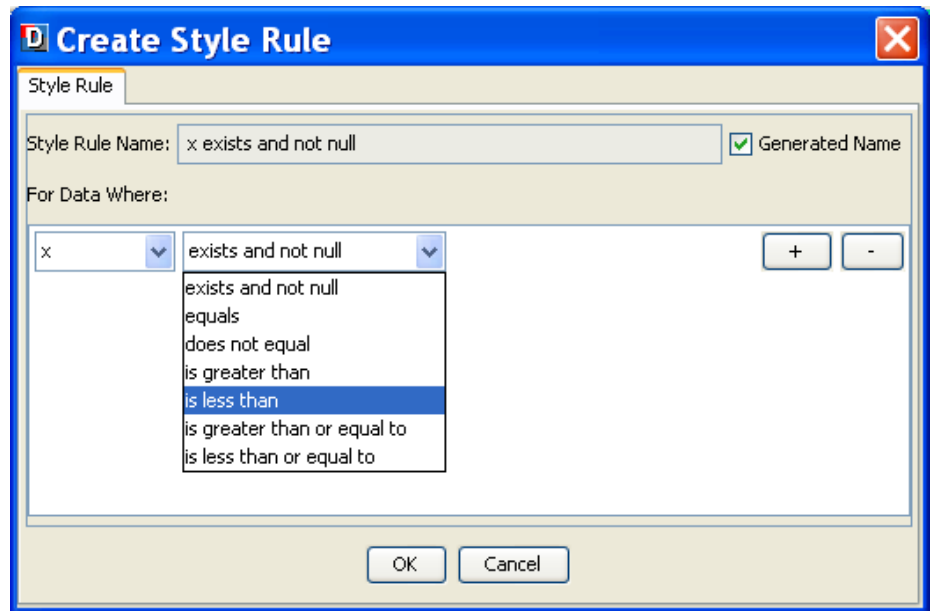
Creating a style rule

To add a new style rule on a data point, proceed as follows:

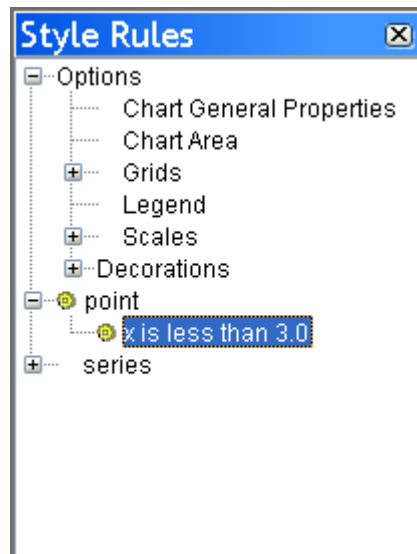
1. Select the option point from the Style Rules tree pane.
2. From the menu bar choose *Edit>Create Style Rule*. The Create Style Rule wizard opens.



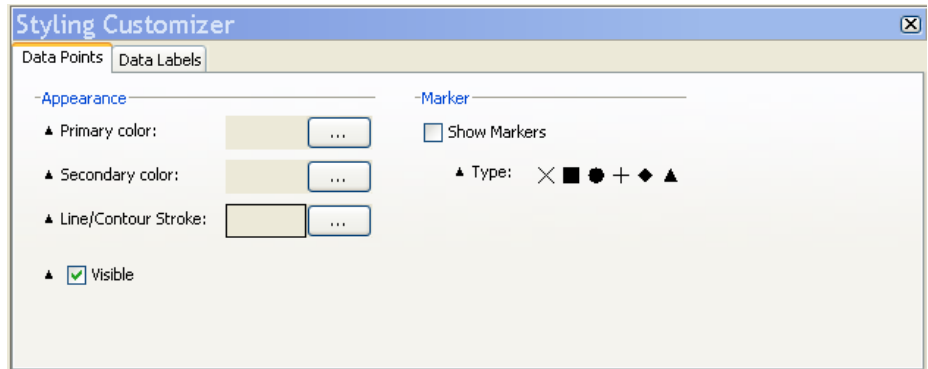
3. Select x from the attributes list.
4. Select “is less than” from the operator menu.



5. An editable drop-down list appears. Enter the value 3.0 and click OK. The new rule appears in the Style Rules tree pane under point.

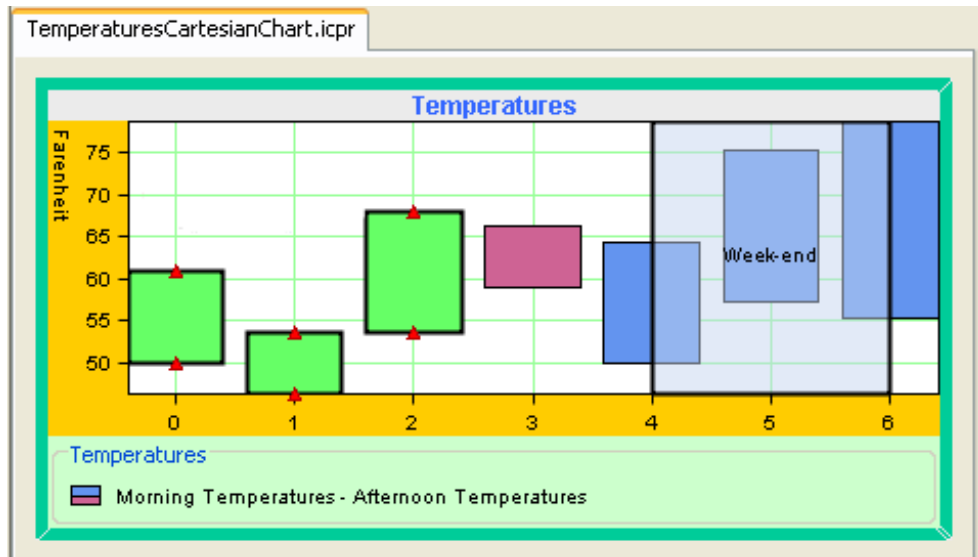


You can customize the new rule by using the Styling Customizer.



6. Choose green as Primary Color.
7. Modify the Line/Contour Stroke attributes by entering 2 in the Line Width field and click Apply.

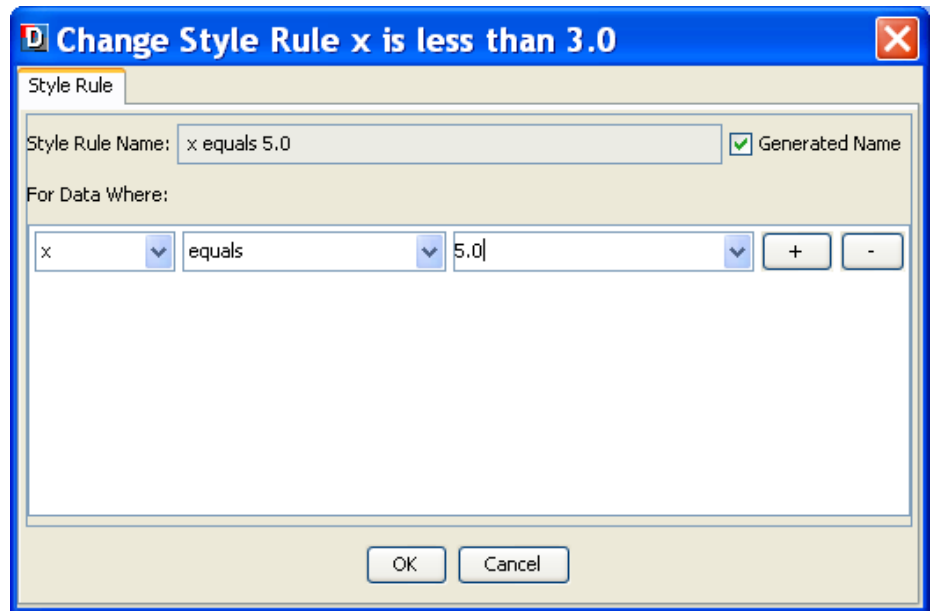
Your chart should now appear as follows:



Modifying an existing style rule

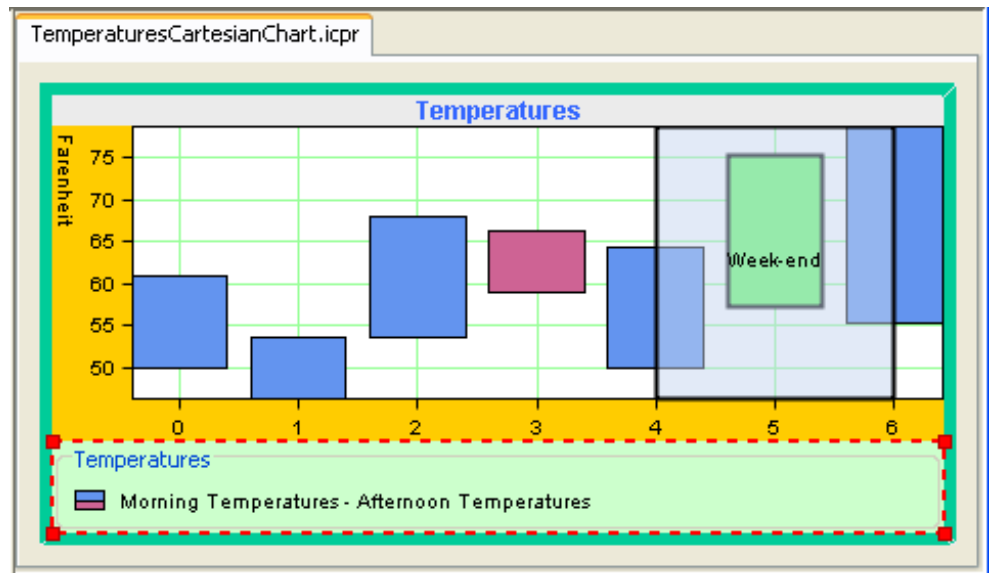
You are going to modify the style rule you have just created.

1. From the Style Rules tree pane select the rule: `point[x is less than 3.0]`.
2. From the menu bar choose *Edit>Change Style Rule*.
3. Modify the attribute as follows: `x equals 5.0`.



4. Click OK to continue.

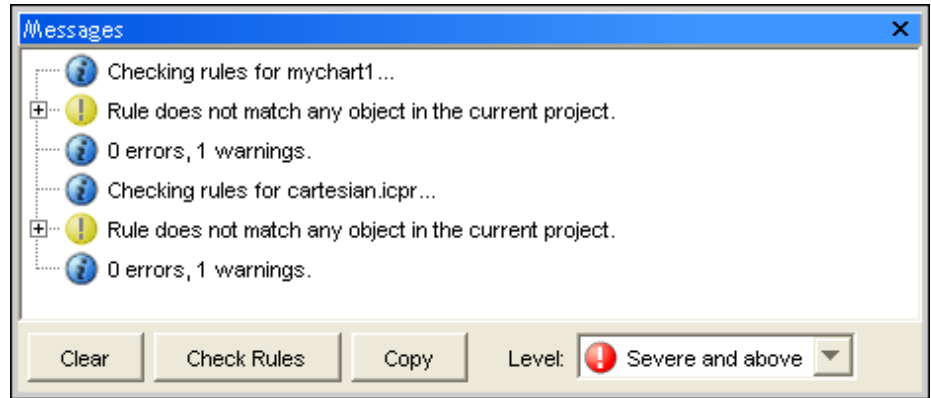
Your chart appears now as follows:



Checking an existing rule

You are going to check the style rule you have just modified.

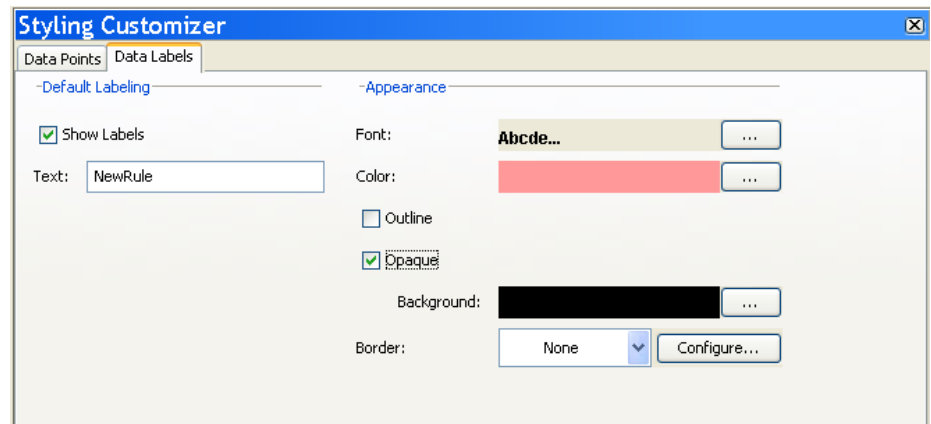
1. From the Style Rules tree pane select the rule `point[x equals 5.0]`.
2. From the menu bar choose *Edit>Check Rules*. The Messages window opens.



The messages help you debug your style rules. For more information, see *Undoing actions*.

Adding data labels

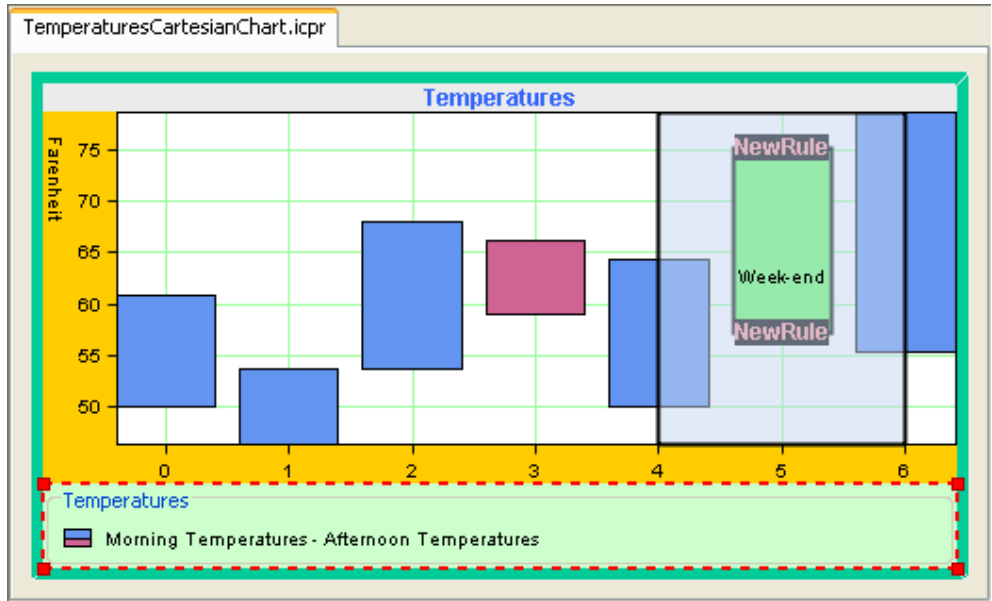
1. With the new rule selected, activate the Data Labels tab in the Styling Customizer.



2. Check the option Show Labels.
3. In the Text field type "NewRule".
4. From the Font editor, change the font properties as indicated below and click Apply:
 - ◆ Font type: Arial

- ◆ Font size: 12
 - ◆ Bold
5. From the Color editor, choose the light red color.
 6. Check the option Opaque.

Your chart appears now as follows:



Data series

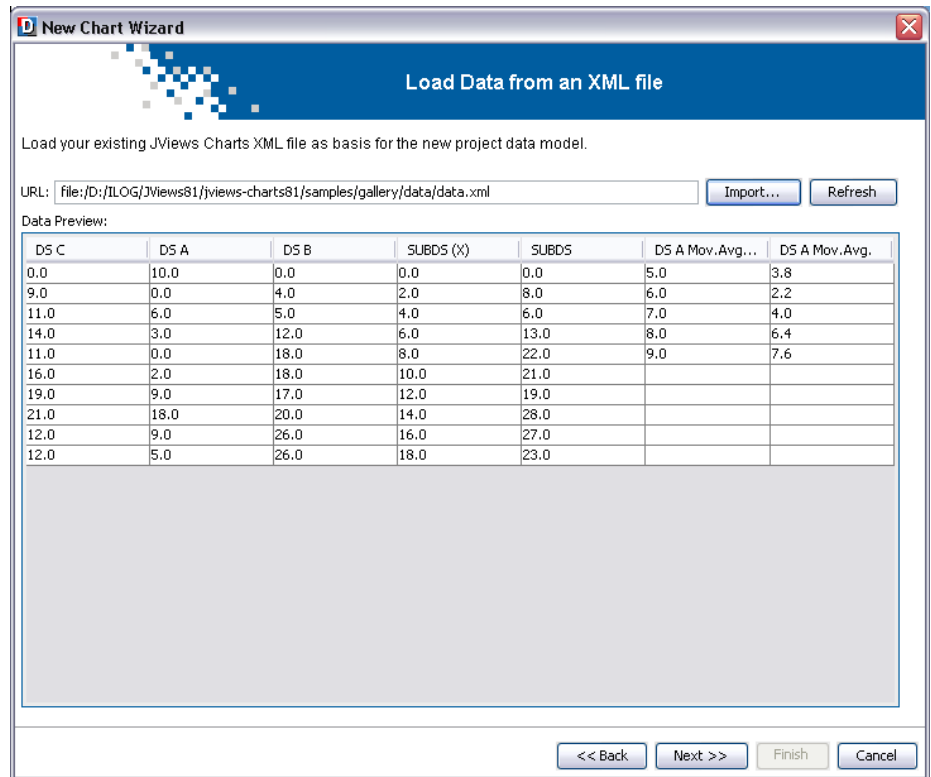
On data series, you can define the following attributes:

- ◆ name: The name of the data set.
- ◆ index: The index of the data set in its data source.

Creating a new Cartesian chart

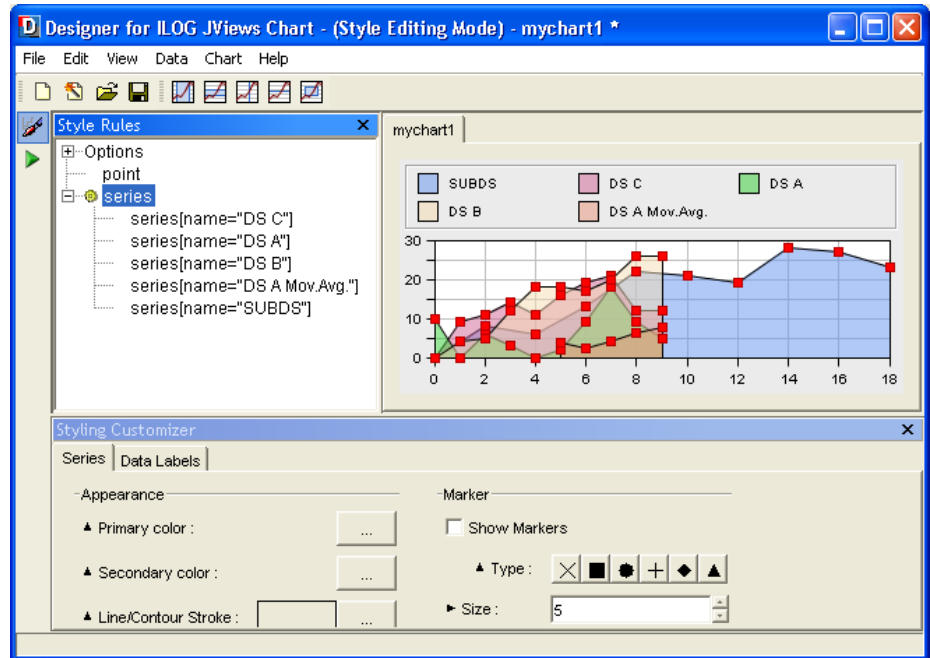
1. Start the New Chart Wizard.
2. Select the option Start From Basic Elements.
3. From the list of data source, choose the XML file.
4. Browse to the directory containing the file `data.xml`.

This file contains a set of series, some being marked with a CSS class. You may look at the file in a text editor to see how the CSS class has been specified.



5. Leave the default mapping and click Next.
6. From the types of chart, choose the Cartesian type.

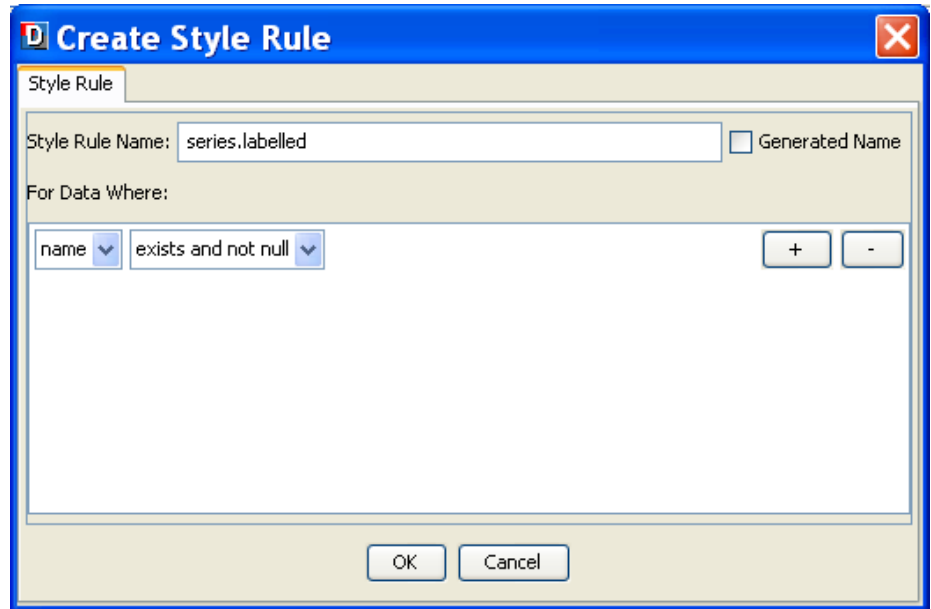
7. From the Representation Family drop-down list choose Area and then select Superimposed.
8. Leave the default theme and click Finish to exit the Wizard.



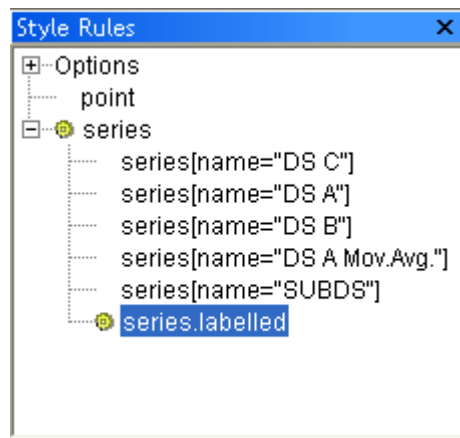
You are now ready to style your data series. The final goal is to display the x and y values as a label annotation for every points of series marked with the 'labelled' CSS class. You will learn how to create a style rule and add data labels.

Creating a style rule on data series

1. Select the option series from the Style Rules tree pane.
2. From the menu bar choose *Edit>Create Style Rule*. The Create Style Rule wizard opens.
3. From the attributes list select "name". From the operator menu select "exists and not null".
4. Uncheck the option Generated Name and type `series.labelled` in the field Style Rule Name. Click OK to confirm.

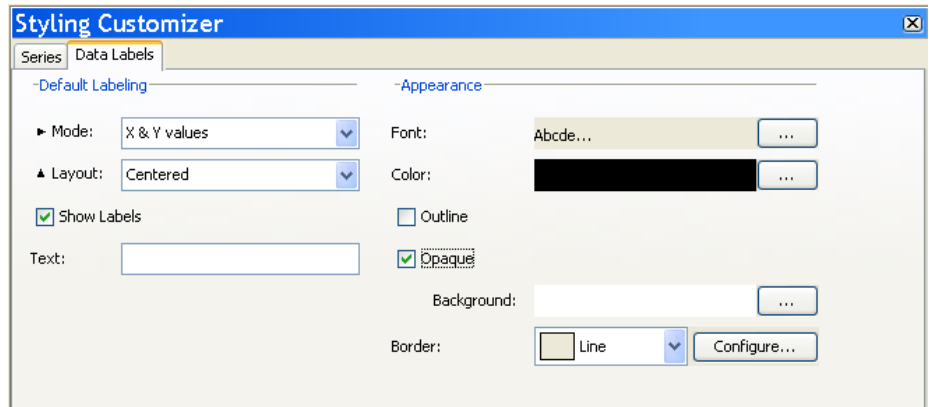


The new rule `series.labelled` appears in the Style Rules tree pane under Series.



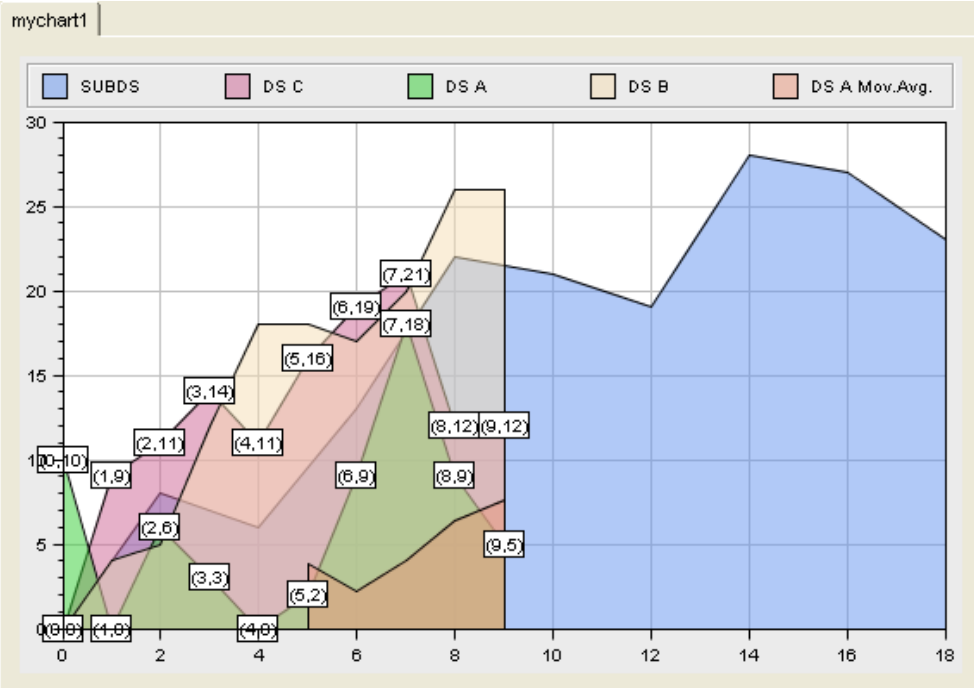
Adding data labels

1. Make sure the Data Labels tab is active in the Style Customizer.



2. Set Mode to X & Y values.
3. Check the option Show Labels.
4. Change the Background color to White.
5. Check the option Opaque.
6. From the Border drop-down list, choose Line.

All the series that have the labelled property are annotated with data labels, as illustrated below:



Using the Rules Menu

You can manage your rules by copying, enabling, disabling, deleting, renaming, and moving rules. To do this, right-click a rule and choose from the Rules Menu. This menu contains the following rule management commands:

Create Style Rule...	
Change Style Rule...	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Paste Styling Properties	
Delete	Delete
Rename	
View	
Move up	
Move down	
Enable	
Disable	

To cut a style rule:

1. Select the rule and right-click.
2. Choose Cut on the Rule Menu. You can undo cut actions, see *Undoing actions*.

To copy a style rule:

1. Select the rule and right-click.
2. Choose Copy on the Rule Menu. Once you have copied a rule, you can do a Paste or Paste Styling Properties action.

To paste:

- ◆ Once you have copied a style rule, select the target one and choose Paste on the Rule Menu. This action changes both the conditions and the styling properties.

To paste a styling rule

- ◆ Once you have copied a style rule, select the target one and choose Paste Styling Properties on the Rule Menu. This action leaves the conditions unchanged and changes the styling properties.

To delete a style rule:

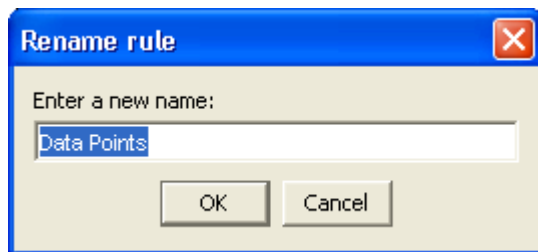
1. Select the rule and right-click.

2. Choose Delete on the Rule Menu. You can undo deletion, see *Undoing actions*.

If you want to customize the appearance of the Style Rules, you can choose your own names for each rule.

To rename a style rule:

- ◆ Select the rule and right-click, then choose Rename and enter your preferred name.



The name you enter appears in place of the dot-separated notation.

You can reorder the style rules within a subtree. Order is important because a style rule that follows others may overwrite property values set by the preceding style rules. The order therefore represents a decision on priority.

Note: Overriding only occurs if the same property is set. If two rules set different properties, their order is unimportant.

To move a style rule up or down:

1. To move a style rule up, select it and right-click, then choose Move up on the Rule Menu.
2. To move a style rule down, select it and right-click, then choose Move down on the Rule Menu.

Note that you cannot place a less specific rule below a more specific rule.

To enable / disable a style rule:

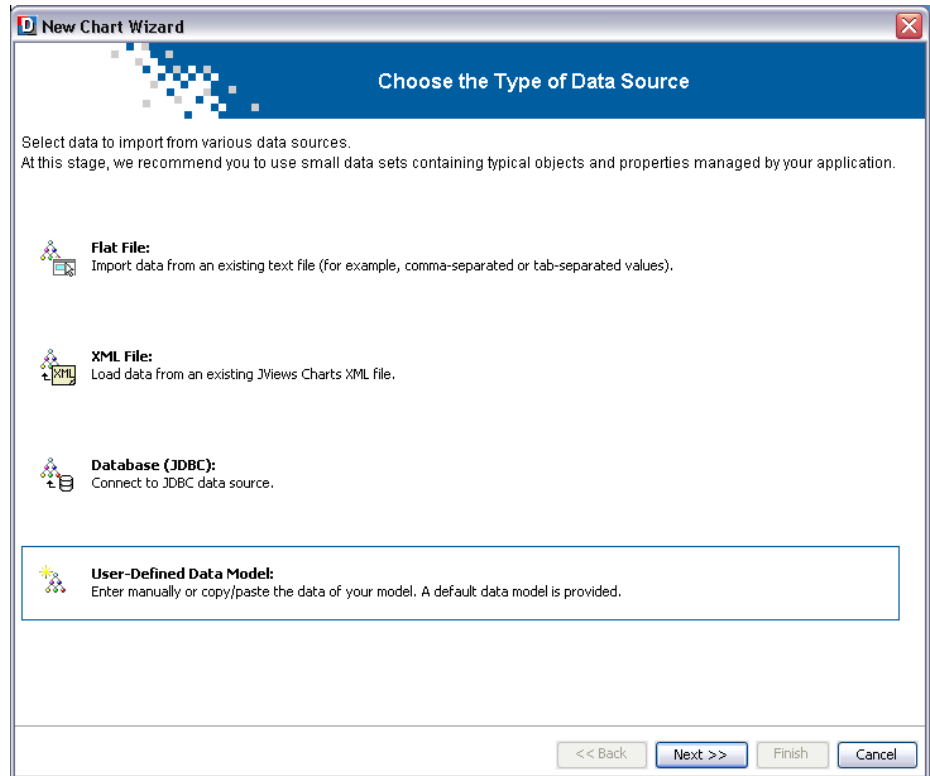
1. To disable a style rule, select it and right-click, then choose Disable on the Rule Menu. A disabled rule appears lighter and in italics.
2. To re-enable a disabled style rule, select it and right-click, then choose Enable on the Rule Menu.

By default, all style rules defined are enabled.

Reconfiguring your data source

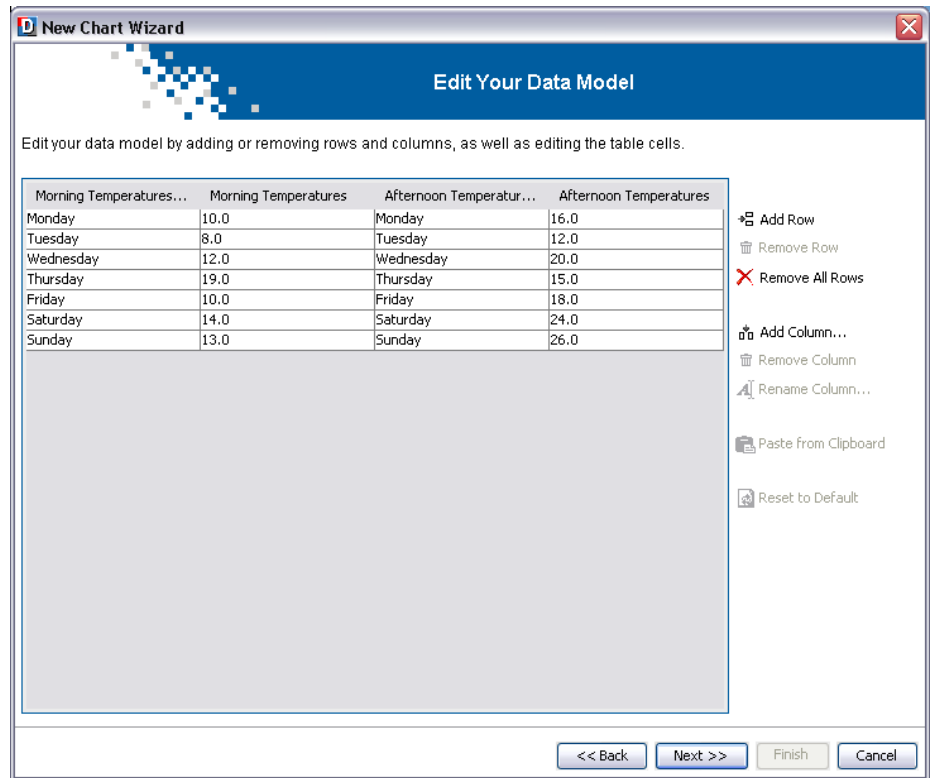
If you have an existing chart, but you want to include further data from the same data source, or try out a new data set, you can reconfigure the data associated with the chart.

1. With your project `TemperaturesCartesianChart.icpr` open, choose *Data>Reconfigure Data*. The New Chart Wizard opens at the Data Source Type page.



2. From the list of data sources, choose User-Defined Data Model.

The window Edit Your Data Model presents the current model connected to the chart as a table. From here, you can add or remove row and columns, clear all the data or reset the model to the default value.



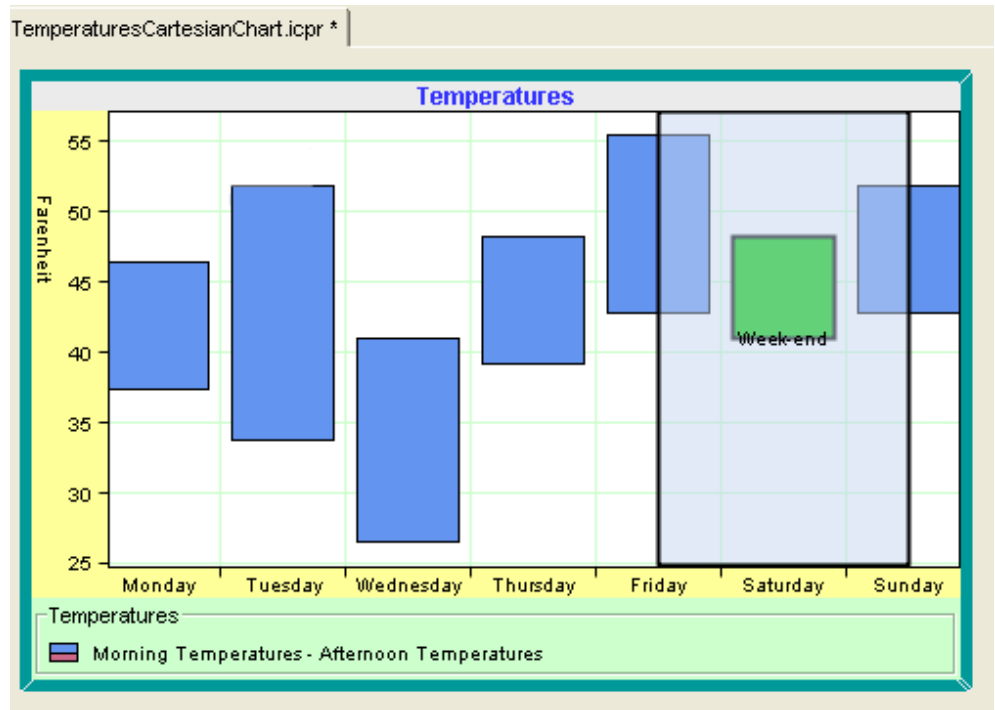
You are going to update the temperatures values with new values. Note that you want to keep the name of the days, so the label columns remain unchanged.

3. Select the first cell of the column Morning Temperatures to change the value from 10 to 3. Press Enter to validate.
4. Select the second cell of the same column to change the value from 10 to 1. Press Enter to validate.
5. Repeat this operation for each cell of the column Morning Temperatures as well as for the cells of the column Afternoon Temperatures to obtain the following data model:

Morning Temperatures	Afternoon Temperatures
3	8
1	11
-3	5
4	9
6	13
5	9
6	11

- Once all the values have been entered, click Next to continue.
- In the window Define the Data Mapping, leave the default mapping and click Finish.

Your chart displays the morning temperatures and afternoon temperatures according to the new values you have just entered.



Undoing actions

The Undo command can apply to the following:




In Style Editing Mode:

- ◆ Property settings
- ◆ Rule changes (creation or modification).

To undo one of these actions, choose *Edit>Undo*. There are unlimited levels of Undo.

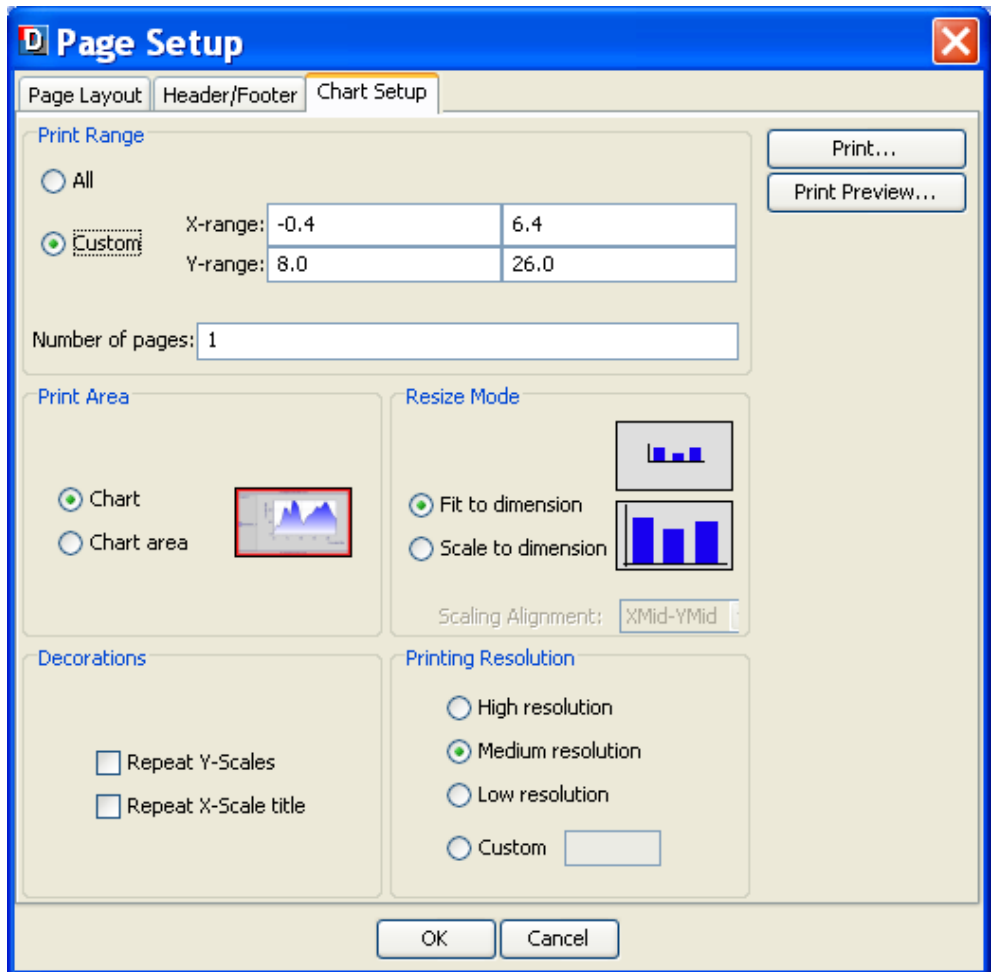
Printing your chart

From the File menu on the menu bar, the following options are available:

- ◆  Page Setup
- ◆  Print Preview
- ◆  Print

Page Setup

From the Chart Setup tab of the Page Setup window, you can define the following settings:



Print Range

The Charts printing framework provides support for multipage printing.

A multipage printing consists of dividing the x-range of the printed data window in successive intervals with respect to the number of pages. You can change the way the data window of a page is computed by checking the option Custom to change the X range and Y range values.

Print Area

Through this option, you can define which chart, or chart area, is printed. Printing a chart means that all the components added to the chart will be printed. That includes the header, the footer, and the legend. On the contrary, printing only the chart area means that all the

other chart components, including the legend, will be ignored. By default, the whole chart is printed.

Resize Mode

You can determine the printed chart size by using the following options:

◆ Fit to dimension.

The chart is expanded to fill the printable area in both dimensions. The original proportions between the chart objects are not preserved.

◆ Scale to dimension.

The chart is expanded in both dimensions proportionally, until the nearest page margins are reached. The original proportions between the chart objects are preserved.

◆ Scale Alignment

This option is active only when the resize mode is set to Scale to dimension. With the Scale Alignment option you specify the value to use when the aspect ratio of the chart does not match the aspect ratio of the printable area. The following figure shows the result for the different values depending on the destination printable area. The default value is XMID_YMID, that is, the chart is centered within the printable area.

Decorations

◆ Repeat Y-Scales

This option sets whether y-scales with a minimum or maximum crossing value (that is, positioned on the axis extremities) need to be printed on each page of the document.

◆ Repeat X-Scale Title

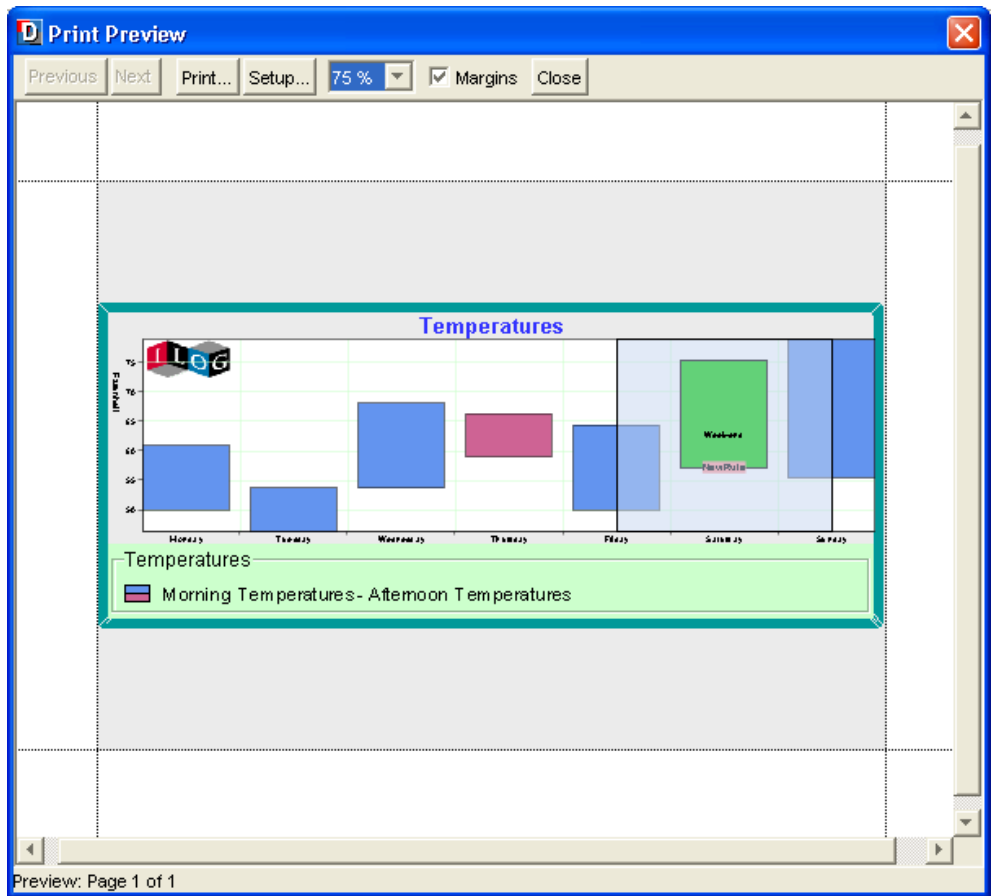
When the x-scale title placement is other than 0 or 100 (that is, the axis extremities), this option sets whether the title has to be printed on each page of the document.

Printing Resolution

Because the printer resolution is usually higher than the screen resolution, there may be some discrepancies between the screen appearance and the printing result. This is particularly true for labels, whose font size may not be adapted for printing. To soften this effect, an additional scale transform may be applied on the printing graphics context when drawing the chart area. The default value is Medium.

Print Preview

Here is the preview of your chart:



Writing an Application

Describes how to integrate your chart component into a Java application and deploy it. This phase includes the writing of some Java code.

In this section

Deployment considerations

Describes the required deployment path and the functional type of the application.

Creating a basic Swing application from a JViews Charts component

Describes the basic steps for creating an application that will use a Chart.

Deployment considerations

In order to build and deploy an application containing an IBM® ILOG® JViews Charts component, you need to first consider the required deployment path and the functional type of the application.

Deployment paths

The deployment paths available are:

- ◆ Swing application
- ◆ Eclipse® /RCP application
- ◆ DHTML thin-client application
- ◆ DHTML-based JSF application
- ◆ Rich Web client application
- ◆ Applet

This section discusses the Swing application path. For discussions of the thin-client deployment path, see *Deploying an application as a DHTML-only thin client* in *Building Web Applications*.

An Eclipse/RCP application with JViews Charts is implemented like a Swing application, with additional considerations explained in *Using JViews products in Eclipse RCP applications*.

For discussions of DHTML-based JSF deployment path, see *Using DHTML-based JSF components to build Web applications* in *Building Web Applications*.

For discussions of the rich Web client deployment path, see *Creating Rich Web Charts* in *Building Web Applications*.

Functional application types

The functional type of the application influences the choice of deployment path.

There are two functional types of application:

- ◆ Display and navigation, providing different views of data but no other interaction
- ◆ Editing, with a graphical editor that provides update capabilities

If you are building a display and navigation application, you can deploy it as a Swing application, a thin client, a rich web client or an applet.

If you are building a graphical editor, you can deploy it as a Swing application or possibly an applet if the user will not need to save a changed version.

Creating a basic Swing application from a JViews Charts component

The Designer essentially allows you to define a configuration for your chart component, without coding it in Java™. While a Designer project file is not mandatory for writing an application using a JViews Charts component, it is a convenient way to easily configure both the data model and the chart component.

1. Create your GUI with various Swing components, such as a working area, a toolbar, and so on.

```
getContentPane().add(button, BorderLayout.NORTH);
```

2. Create an instance of a Chart in the Java API.

```
// Creates the chart instance on which project file will be applied.
chart = new IlvChart();
getContentPane().add(chart, BorderLayout.CENTER);
```

3. Connect the instance to a data model or load a project file that has been created and saved in the Designer. In this example, we connect a project file that is chosen from a File Browser shown when the button is pressed.

```
JFileChooser fileChooser = new JFileChooser();
...
int ret = fileChooser.showOpenDialog(ProjectViewer.this);
if (ret == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();
    try {
        URL project = file.toURI().toURL();
        // Apply the project file to the chart
        chart.setProject(project);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

See *Integrating your development into an application* for more about loading a project file.

Instead of loading a project file, you can load a custom data model and then load a style sheet (.css file) defined in the Designer. In this case, you use only the styling developed with the Designer and not the whole project.

Refer to the samples provided with IBM® ILOG® JViews Chart for examples on how to create applications that use charts.

The complete sample `ProjectViewer.java` is located in:

```
<installdir>/jviews-charts86/codefragments/chart/project-viewer/src/  
ProjectViewer.java
```

Refer to *Developing with the JViews Charts SDK* to see how to create a chart and customize it with the SDK instead of the Designer.

XML File Format

Describes the XML structure and the XML project file.

In this section

XML Structure

Describes the schema and the Document Type Definition for XML data file, the XML Elements and the Interpretation of Values.

XML project file

Describes the format of an IBM® ILOG® JViews Charts project file.

XML Structure

Describes the schema and the Document Type Definition for XML data file, the XML Elements and the Interpretation of Values.

In this section

Schema for XML data file

Describes the XML elements used to describe data by reference to the file temperatures.xml.

Document Type Definition for XML data file

Describes the Document Type Definition (DTD) that presents the Charts XML data file syntax.

XML elements

Describes the XML elements used in the example, their attributes, and subelements together with their attributes and subelements.

Interpretation of values

Explains some of the values stored in XML data files.

Schema for XML data file

For more details on how the Designer loads data from the `temperatures.xml` file, see *Loading your data*.

The XML file used in the examples in *Getting Started* is located in:

```
<installdir>/jviews-charts86/samples/gallery/data/temperatures.xml
```

For more information, you can also refer to Reading and writing data from an XML source in *Developing with the SDK*.

This is the XML schema that describes the Charts XML data file syntax. You can find the XML schema in the file:

```
<installdir>/jviews-charts86/data/chartxml.xsd
```

```
public ProjectViewer()
{
    super("Viewer of Designer for ILOG JViews Charts project files");
    JButton button = new JButton("Open Project File");
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            ...
        }
    });
}
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="chartData">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="data" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="xmlns:ilvchart" use="fixed"
value="http://www.ilog.com/products/jviews/chart" type="xsd:string"/>
      <xsd:attribute name="version" use="required" type="xsd:string"/>
      <xsd:attribute name="xml:lang" type="xsd:NMTOKEN"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="data">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="labels" minOccurs="0" maxOccurs="1"/>
        <xsd:element ref="series" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="xSeries" type="xsd:IDREF"/>
      <xsd:attribute name="xml:lang" type="xsd:NMTOKEN"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="series">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:choice>
          <xsd:element ref="valueOperator" minOccurs="1" maxOccurs="1"/>

```

```

        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="value"/>
            <xsd:element ref="valuesList"/>
        </xsd:choice>
    </xsd:choice>
    <xsd:element ref="labels" minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="property" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="dateFormat" type="xsd:string"/>
<xsd:attribute name="type" use="required">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="double"/>
            <xsd:enumeration value="date"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
<xsd:attribute name="name" type="xsd:string"/>
<xsd:attribute name="id" use="required" type="xsd:ID"/>
<xsd:attribute name="xml:lang" type="xsd:NMTOKEN"/>
</xsd:complexType>
</xsd:element>
<xsd:element name="labels">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element ref="label" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="delimiter" type="xsd:string"/>
        <xsd:attribute ref="xml:space" default="preserve"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="value">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="xml:lang" type="xsd:NMTOKEN"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="label">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute ref="xml:space" default="preserve"/>
                <xsd:whiteSpace value="preserve"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
<xsd:element name="valuesList">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">

```

```

        <xsd:attribute name="delimiter" type="xsd:string"/>
        <xsd:attribute name="xml:lang" type="xsd:NMTOKEN"/>
    </xsd:extension>
</xsd:simpleContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="valueOperator">
    <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element ref="seriesRef"/>
            <xsd:element ref="property"/>
        </xsd:choice>
        <xsd:attribute name="class" use="required" type="xsd:NMTOKEN"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="seriesRef">
    <xsd:complexType>
        <xsd:attribute name="ref" use="required" type="xsd:IDREF"/>
    </xsd:complexType>
</xsd:element>
<xsd:element name="property">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="name" use="required" type="xsd:string"/>
                <xsd:attribute name="javaClass" type="xsd:string"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
</xsd:schema>

```

Document Type Definition for XML data file

You can find the DTD in the file:

<installldir>/jviews-charts86/data/chartxml.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!ELEMENT chartData (data+)>
<!ATTLIST chartData xmlns:ilvchart CDATA #FIXED
"http://www.ilog.com/products/jviews/chart"
        version CDATA #REQUIRED
        xml:lang NMTOKEN #IMPLIED>

<!ELEMENT data (labels?, series+)>
<!ATTLIST data xSeries IDREF #IMPLIED
        xml:lang NMTOKEN #IMPLIED>

<!ELEMENT series ((valueOperator | (value | valuesList)*), labels?, property*)
>
<!ATTLIST series dateFormat CDATA #IMPLIED
        type (double | date) #REQUIRED
        name CDATA #IMPLIED
        id ID #REQUIRED
        xml:lang NMTOKEN #IMPLIED>

<!ELEMENT labels (#PCDATA | label)*>
<!ATTLIST labels delimiter CDATA #IMPLIED
        xml:space (preserve) #FIXED 'preserve'>

<!ELEMENT value (#PCDATA)>
<!ATTLIST value xml:lang NMTOKEN #IMPLIED>

<!ELEMENT label (#PCDATA)>
<!ATTLIST label xml:space (preserve) #FIXED 'preserve'>

<!ELEMENT valuesList (#PCDATA)>
<!ATTLIST valuesList delimiter CDATA #IMPLIED
        xml:lang NMTOKEN #IMPLIED>

<!ELEMENT valueOperator (seriesRef|property)*>
<!ATTLIST valueOperator class NMTOKEN #REQUIRED>

<!ELEMENT seriesRef EMPTY>
<!ATTLIST seriesRef ref IDREF #REQUIRED>

<!ENTITY % propertyExt "">
<!ELEMENT property (#PCDATA %propertyExt;)*>
<!ATTLIST property name CDATA #REQUIRED
        javaClass CDATA #IMPLIED>
```

XML elements

Elements are enclosed in angle brackets (< >) in the code in accordance with XML conventions. The content dependent on each element must be terminated by an end tag denoted by a diagonal or solidus (/) followed by the same element name as the start element. For example:

```
<series> ... </series>
```

For a <property> element, you must give the name of the property ("propertyname") and its value. The syntax is:

```
<property name="propertyname">value</property>
```

This section describes the following XML elements:

- ◆ *The chartData Element*
- ◆ *The data Element*
- ◆ *The series Element*
- ◆ *The valuesList Element*
- ◆ *The value Element*
- ◆ *The valueOperator Element*
- ◆ *The property Element*
- ◆ *The labels Element*
- ◆ *The label Element*
- ◆ *The ?xml-stylesheet? Processing Instruction*

The chartData Element

This element is the root element of the XML file. It represents an entire `IlvDataSource`.

Example

```
<chartData version="1.0">
  <data>
    <series id="Morning_Temperatures" name="Morning Temperatures"
      type="double">
      <valuesList delimiter=",">10.0,8.0,12.0,19.0,10.0,14.0,13.0</valuesList>
      <labels delimiter=",&#10;&#13;">Monday,Tuesday,Wednesday,Thursday,
        Friday,Saturday,Sunday</labels>
    </series>
  </data>
</chartData>
```

```

<series id="Afternoon_Temperatures" name="Afternoon Temperatures"
  type="double">
  <valuesList delimiter=",">16.0,12.0,20.0,15.0,18.0,24.0,26.0</valuesList>

  <labels delimiter=","&#10;&#13;">Monday,Tuesday,Wednesday,Thursday,
    Friday,Saturday,Sunday</labels>
</series>
</data>
</chartData>

```

chartData Element

Element	Attribute	Default	Description
<chartData>			An entire <code>IlvDataSource</code> . Count: 1. Child elements: <data>
	version	required	Version of the ChartXML. Currently "1.0".

The data Element

The <data> element represents one or more `IlvDataSets`. Typically it represents exactly one `IlvDataSet`.

Each <data> element can be seen as a table in which each <series> element represents one or two columns. One of these series can be identified as the one holding the x-values. In this case, the other series are assumed to hold the y-values of the considered data.

Series data points might be associated with labels by means of the <labels> element. Labels can be either defined common to all the series of a data element, or individually for each series. In this case, the labels are specified at the <series> element level, and override labels that may have been set on the <data> element.

Example

```

<data>
  <series id="Morning_Temperatures" name="Morning Temperatures"
    type="double">
    <valuesList delimiter=",">10.0,8.0,12.0,19.0,10.0,14.0,13.0</valuesList>
    <labels delimiter=","&#10;&#13;">Monday,Tuesday,Wednesday,Thursday,
      Friday,Saturday,Sunday</labels>
  </series>
  <series id="Afternoon_Temperatures" name="Afternoon Temperatures"
    type="double">
    <valuesList delimiter=",">16.0,12.0,20.0,15.0,18.0,24.0,26.0</valuesList>

    <labels delimiter=","&#10;&#13;">Monday,Tuesday,Wednesday,Thursday,
      Friday,Saturday,Sunday</labels>

```

```
</series>
</data>
```

data Element

Element	Attribute	Default	Description
<data>			One or more <code>IlvDataSets</code> . Child elements: <series>, <labels>.
	xSeries		Represents the ID of a series that contains the x values of the dataset(s) denoted by this <data> element. The other series elements are y values.

The series Element

This element specifies a set of values (either X or Y) in an `IlvDataSet`.

Each series is identified by a unique ID and the type of data it contains (either double values or dates). The `dateFormat` attribute can be any pattern that conforms to the syntax used by the `java.text.SimpleDateFormat` class. Values can be expressed with elementary <value> elements, or as a list with the <valuesList> element.

Example

```
<series id="Morning_Temperatures" name="Morning Temperatures"
  type="double">
  <valuesList delimiter=",">10.0,8.0,12.0,19.0,10.0,14.0,13.0</valuesList>
  <labels delimiter=",&#10;&#13;">Monday,Tuesday,Wednesday,Thursday,
    Friday,Saturday,Sunday</labels>
</series>
```

series Element

Element	Attribute	Default	Description
<series>			An ordered list of values. Child elements: <valuesList>, <value>, <valueOperator>, <property>, <labels>.
	id	required	Identifies this element uniquely within the XML file. Used for other series that need to refer to this element.
	type	required	Element type of the series: either date or double.
	dateFormat	dd-MMMM-yy	For series of type date, the format of the elements in this XML element is given by this <code>dateFormat</code> attribute. For

Element	Attribute	Default	Description
			the interpretation of this string, refer to <code>java.text.SimpleDateFormat</code> .
	<code>name</code>		The user-visible name of this list of values.

The valuesList Element

This element represents an ordered list of values. Their type is given by the `type` attribute of the enclosing `<series>` element. They are separated by delimiters or white space. The default set of delimiters is only a comma. For numbers, the decimal point is a dot, regardless of the locale. For dates, the format is given by the `dateFormat` attribute of the enclosing `<series>` element.

Example

```
<valuesList delimiter=",">10.0,8.0,12.0,19.0,10.0,14.0,13.0</valuesList>
```

valuesList Element

Element	Attribute	Default	Description
<code><valuesList></code>			An ordered list of values.
	<code>delimiter</code>	<code>“,”</code>	Set of delimiters (other than white space) that separate the values from each other.

The value Element

This element represents a single value. Its type is given by the `type` attribute of the enclosing `<series>` element. For numbers, the decimal point is a dot, regardless of the locale. For dates, the format is given by the `dateFormat` attribute of the enclosing `<series>` element.

Example

```
<value>19.0</value>
```

value Element

Element	Attribute	Default	Description
<code><value></code>			A single value.

The valueOperator Element

This element represents a list of values, computed from the list of values of another `<series>` element.

Example

```
<valueOperator class="com.foo.Exponential">
  <seriesRef ref="x"/>
</valueOperator>
```

valueOperator Element

Element	Attribute	Default	Description
<valueOperaotr>			Computed list of values. Child Elements: <seriesRef>, <property>.
	class	required	The Java™ class name of a subclass of <code>IlvDataSet</code> .

The seriesRef Element

This element represents a symbolic reference to a <series> element.

Example

```
<seriesRef ref="x"/>
```

seriesRef Element

Element	Attribute	Default	Description
<seriesRef>			Reference to a <series> element.
	ref	required	The ID of the series being referenced.

The property Element

This element represents a property of the enclosing element. In other words, it is a key/value pair, where the key is the name of the property and the value is its contents.

Example

```
<property name="period">20</property>
```

property Element

Element	Attribute	Default	Description
<property>			A named property.
	name	required	The name of the property.
	javaClass		For CSS properties, this specifies the Java class of the property value.

The labels Element

This element represents the labels of a series. The labels are in the same order as the data values. They can be specified either as a list of <label> elements, or as the text contents of the <labels> element. In this case, they are separated by delimiters. The default set of delimiters is only a comma. Note that none of the labels can contain one of the delimiter characters.

Example

The first possible notation is as follows:

```
<labels>
<label>Monday</label>
<label>Tuesday</label>
<label>Wednesday</label>
<label>Thursday</label>
<label>Friday</label>
<label>Saturday</label>
<label>Sunday</label>
</labels>
```

The second possible notation is as follows:

in this example we have broken the element in two lines, and therefore added the newline and space to the set of delimiter characters.

```
labels delimiter=" ,&#10;&#13; " >Monday,Tuesday,Wednesday,Thursday,
    Friday,Saturday,Sunday</labels>
```

labels Element

Element	Attribute	Default	Description
<labels>			Labels of a series. Child elements: <label>
	delimiter	“,”	Set of delimiters that separate the labels from each other.

The label Element

This element represents a single label.

label Element

Element	Attribute	Default	Description
<label>			A single label.

The ?xml-stylesheet? Processing Instruction

Specifies a stylesheet to be used with the data source. It contains a `href` attribute containing an URL or a relative file name. It is possible to specify this processing instruction more than once; the style sheets are then cascaded (cumulated).

Example

```
<?xml-stylesheet type="text/jviews+css" href="styles/chart1.css"?>
```

Interpretation of values

Some of the values stored in XML data files, such as dates, are represented in a particular locale. For example, "28-May-73" is a valid representation of a date in an English locale, but not in a French or Japanese locale.

The XML elements `<chartData>`, `<data>`, `<series>`, `<valuesList>`, `<value>` can contain an attribute `'xml:lang'`. The value of this attribute is the name of a locale, such as "en-US" for English in the U.S., "ja" for Japanese, or "zh-Hans" for Simplified Chinese. This attribute applies to all values inside this element. An `'xml:lang'` attribute on an element overrides the one inherited from the parent element. For further details, see <http://www.w3.org/TR/REC-xml/#sec-lang-tag> and <http://www.w3.org/International/articles/language-tags/Overview.en.php>.

When a Charts XML data file is read, the default locale - applied to values that are not inside an element that carries an `'xml:lang'` attribute - is the locale of the Java VM.

Since JViews 8.0 patch 3, when a Charts XML data file is saved by the Charts Designer, an `'xml:lang'` attribute is attached to the `<chartData>` element.

Earlier versions saved files without such an attribute. In your application, you can customize the locale used, and whether to use an `'xml:lang'` attribute at all, through the method `setLocale(java.util.Locale)`.

XML project file

The IBM® ILOG® JViews Charts project files are identified by a filename ending in ".icpr". A project file is an XML file. An example file is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<chartConfiguration xmlns:xlink="http://www.w3.org/1999/xlink">
  <style xlink:href="myproject.css"/>
  <dataSource type="ilog.views.chart.data.IlvDefaultDataSource">
    <model>
      <chartData version="1.0">
        ...
      </chartData>
    </model>
  </dataSource>
</chartConfiguration>
```

There is no schema or DTD for this file format, because some parts can be extended in arbitrary ways by defining particular Java classes.

XML Elements

The following tables describe the XML elements used in the example, their attributes, and subelements together with their attributes and subelements. Elements are enclosed in angle brackets (< >) in the code in accordance with XML conventions. The content dependent on each element must be terminated by an end tag denoted by a diagonal or solidus (/) followed by the same element name as the start element. For example:

```
<chartData> ... </chartData>
```

For a <property> element, you must give the name of the property ("propertyname") and its value. The syntax is:

```
<property name="propertyname">value</property>
```

This section describes the following XML elements:

- ◆ *The chartConfiguration Element*
- ◆ *The style Element*
- ◆ *The dataSource Element*

The chartConfiguration Element

This element is the root element of a Charts project file. It represents a chart together with its data source and styling.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<chartConfiguration xmlns:xlink="http://www.w3.org/1999/xlink">
  <style xlink:href="myproject.css"/>
  <dataSource type="ilog.views.chart.data.IlvDefaultDataSource">
    <model>
      <chartData version="1.0">
        ...
      </chartData>
    </model>
  </dataSource>
</chartConfiguration>
```

The *chartConfiguration* Element

Element	Attribute	Default	Description
<chartConfiguration>			An entire IlvChart. Count: 1. Child elements: <style>, <dataSource>

The style Element

This element represents some styling rules for a chart. Several <style> elements can appear. In this case, the corresponding CSS files are implicitly combined (cascaded).

Example

```
<style xlink:href="myproject.css"/>
```

The *style* Element

Element	Attribute	Default	Description
<style>3			A file with CSS rules. Count: 0 or more.
	xlink:href	required	URL of a CSS file. If it is a relative URL, it is interpreted as being relative to the project file.

The dataSource Element

This element represents the data source connected to the chart.

Example

```
<dataSource type="ilog.views.chart.data.IlvDefaultDataSource">
```

```

<model>
  <chartData version="1.0">
    ...
  </chartData>
</model>
</dataSource>

```

The dataSource Element

Element	Attribute	Default	Description
<dataSource>			A data source. Count: 0 or 1. Child elements: depend on the type attribute.
	type	required	Java class name of the data source.

The data source class must implement the `IlvDataSource` interface and must have a `IlvChartConfiguration.IlvDataSourceXMLSerializer` instance registered via `IlvChartConfiguration.setDataSourceSerializer`.

A data source serializer is already predefined for the following classes: `IlvXMLDataSource`, `IlvJDBCDataSource`, `IlvDefaultDataSource`, `IlvSwingTableDataSource`. The contents of the <dataSource> element depends on this class.

A <dataSource> element of the type `IlvXMLDataSource` has the following child elements:

Element	Attribute	Default	Description
<file>			A reference to a file containing a data source. See <i>XML File Format</i> . Count: 1.
	location	required	The URL of the data file.

A <dataSource> element of the type `IlvDefaultDataSource` or `IlvSwingTableDataSource` has the following child elements:

Element	Attribute	Default	Description
<model>			A data source. Count: 1. Child elements: <chartData>
<chartData>			Data. Count: 1 See <i>XML File Format</i> .

A <dataSource> element of the type `IlvJDBCDataSource` has the following child elements:

Element	Attribute	Default	Description
<connection>			The connection to the database. Count: 1.
	param	required	All connection parameters, packed together.
<mapping>			Mapping from columns to a data set..
	xSeries		Name of the column containing the x-values.
	ySeries	required	Names of the columns containing the y-values, as a comma-separated list of column names.
	labels		Name of the column containing the x-labels.

Next Steps After the Designer

Explains how you can use the SDK and the Designer together to customize or extend your chart application.

In this section

Integrating the Project File Into an Application

Explains how to create an application that shows the same chart you created in the Designer.

Integrating the styling into an application

Explains how to create an application that shows data from an arbitrary data source, styled with a `.css` file that was created with the Designer.

Styling: CSS versus API

This section explains the relationship between CSS styling and Java API.

Integrating the Project File Into an Application

This is particularly useful when the data to be displayed belongs to one of the predefined data source types.

The result of working with the Designer is a JViews Charts project file with the extension `.icpr` and a CSS style sheet with the extension `.css`. The project file contains a reference to the style sheet.

Using a project file with a predefined data source

- ◆ Integrate the data source and the styling into the chart by busing the following code:

Integrating Data Source and Styling From the Project File

```
IlvChart chart = new IlvChart();
URL project = (new File("myproject.icpr")).toURL();
chart.setProject(project);
```

A complete example can be found in `<installdir>/jviews-charts86/codefragments/chart/project-viewer/src/ProjectViewer.java`.

Customizing a chart loaded from a project file

It is possible to customize a chart loaded from a project file.

- ◆ Customize your chart by adding interactors:

```
chart.addInteractor(my3DInteractor);
```

A complete example can be found in `<installdir>/jviews-charts86/codefragments/chart/project-based/src/ProjectBased.java`.

Integrating the styling into an application

This allows you to use a different data source. The `.icpr` file is not used in this approach.

- ◆ To do this, set the data source and the styling separately.

Setting the Data Source and the Styling Separately

```
IlvChart chart = new IlvChart();
IlvDataSource myDataSource = ...;
chart.setDataSource(myDataSource);
chart.setStyleSheet("myproject.css");
```

A complete example can be found in `<installdir>/jviews-charts86/codefragments/chart/styling/src/ChartCSSExample.java`.

See *Extending the Data Model* in *Developing with the SDK* for how to develop a custom data source.

It is a good idea to run the Designer on a snapshot of real data from your data source. Once you have implemented your custom data source, you can import data into the Designer by following the procedure explained in *Reading and writing data from an XML source* in *Developing with the SDK*, namely by using `IlvXMLDataWriter` to create an XML file with snapshot data.

Styling: CSS versus API

The CSS styling allows you to change many graphical parameters. The Java™ API of JViews Charts also allows you to change graphical parameters. In the following sections you are going to see what their peculiarities are and how they can be used together.

Relationship Between CSS and API

The CSS uses the JavaBeans™ setter methods of `IlvChart` and its associated objects. Therefore, what can be implemented through CSS can also be implemented through explicit API calls. If you use both of them, you have to be aware that loading a CSS file erases all the Bean property values that were set since the CSS was last applied. This means that the API changes made after loading a CSS file will override the CSS settings; however, it is not recommended to reload a CSS file after applying API changes to the same properties.

Available Graphical Parameters

All graphical parameters that are available through the CSS are also available in the API. Some customizations are only available through the API; this is the case for the treemap renderer. Moreover, the API allows you to use your own custom instances of `Paint`, `Stroke`, and `IlvStyle`, while the CSS offers a broad but predefined set of parametrizations of `Paint`, `Stroke` and `IlvStyle`.

Computed Graphical Parameters

When the values of graphical parameters depend on other variables in your application, you may need to define CSS functions (subclasses of `IlvCSSFunction`) and register them through `IlvChart.registerFunction`, so that they can be used in CSS style sheets. Since this is not immediate to do, it might be easier to do the corresponding styling in Java.

Ease of Development

The Designer and the Styling Customizer allow you quickly try out various parameters (or combination of parameters) until you obtain the desired graphical effect. It might be more tedious to do this at the API level.

If you decide to customize your chart through the API rather than through CSS, you can still make limited use of the Designer to implement graphical effects. You can use the Designer to discover the right graphical parameters and then transpose them from CSS to Java code. For example, if the CSS is as follows

```
chartArea {
    foreground : black;
}
```

the API call will be:

```
chart.getChartArea().setForeground(Color.black);
```

Debugging

Debugging CSS is not easy. Sometimes a CSS rule hides part of another CSS rule, or you may not be aware of some of the complexities of CSS processing. On the other hand, styling code written in Java is more immediate to understand and can be debugged with the usual Java IDE features (such as breakpoints).

Interactions

Interactions are usually defined through an interactor class. Since you cannot test it in the Designer environment, you can usually add such an interactor class through the API, not through CSS.

Index

- A**
 - application
 - testing **21**
- B**
 - basic
 - creating a Cartesian chart **10**
 - elements **10**
- C**
 - chart
 - selecting the type **14**
 - ChartCSSExample.java sample **137**
 - charts
 - Cartesian **10**
- D**
 - data
 - loading **11**
 - mapping **13**
 - Data Range **77**
 - data source
 - choosing **11**
 - decorations **81**
 - Designer
 - Running the **9**
 - drawing order **81**
- I**
 - IlvXMLDataWriter class **137**
 - IlvXMLDataWriter.setLocale(java.util.Locale)
method **129**
- L**
 - loading
 - from XML file **11**
- P**
 - ProjectBased.java sample **136**
 - ProjectViewer.java sample **22, 113, 136**
- S**
 - Startup **9**
- T**
 - theme
 - choosing **17**
- U**
 - UNIX **9**
- V**
 - Visible Range **77**
- W**
 - Windows **9**