IBM ILOG JViews Gantt V8.6

Introducing JViews Gantt

Table of contents

How JViews Gantt can help you

Describes the different features furnished by JViews Gantt, the added value these features give to you and their typical uses.

In this section

Charts for planning and scheduling

Time and activity charts most commonly take the form of a bar chart. The main purpose of such a chart is to represent the relationship of events, activities, or actions to time by using combinations of words, numbers, and graphics to represent activities and events. Time is displayed on the horizontal axis.

Activities or resources are displayed on the vertical axis. Resources might be people, places, or things related to performing the activities.

Blocks of time are delimited by using horizontal bars, colors, symbols, and so forth.

The relationship between activities and time can also be displayed in the form of a calendar view. The main purpose of such a view is to represent the relationship of activities to the days of a month or to the working hours in a day. Blocks of time are delimited by overlaying horizontal bars and symbols on top of a calendar or hourly grid.

Typical uses of time and activity charts are for:

- ♦ Scheduling
- ♦ Loading
- ♦ Project planning
- ♦ Managing activities

Time and activity charts are used to communicate information in these domains in a way that is easy to assimilate because of its graphic representation.

The size and complexity of these charts range from small scheduling charts used to plan office-based tasks to very complicated charts used for major projects that require updating by computer.

There are specific types of time and activity chart, but these types of chart are often customized to fit user requirements.

Time and activity bar charts are frequently used to plan activities, such as projects, that have a distinct start and end. This type of chart is often called a *Gantt chart*.

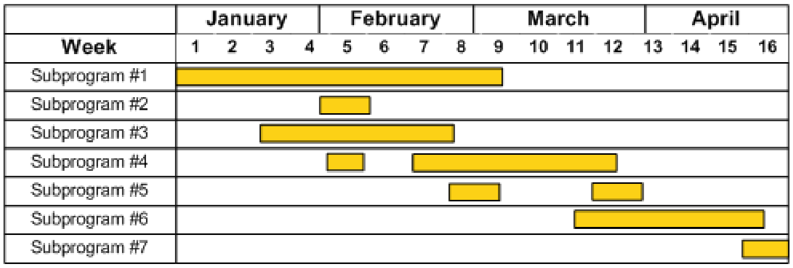
A Gantt chart is a time and activity bar chart that is used for planning, and controlling projects or programs that have a distinct beginning or end. In a Gantt chart, each main activity that is involved in the completion of the overall project or program is represented by a horizontal bar.

The ends of the bar represent the start and end of the activity. The start and end times may be estimated times if the chart is used for planning purposes only. When a chart is used for planning and tracking, the start and end times of future activities are estimated. The historical parts of the chart show the real start and end times of completed activities.

Significant information can be indicated by the bars in addition to marking the real or projected time intervals. For example, the bars can show the difference between elapsed time and worked time, the major steps or phases of an activity, or the internal or external resourcing of an activity.

When an activity is estimated to take a considerable amount of time, it can be broken down into smaller chunks. The completion of each chunk is recorded as a milestone. A specific visual representation is associated with milestones. The chunking of an activity and the use of milestones provide closer control and earlier detection of slippages or other difficulties. This type of Gantt chart is sometimes called a milestone chart.

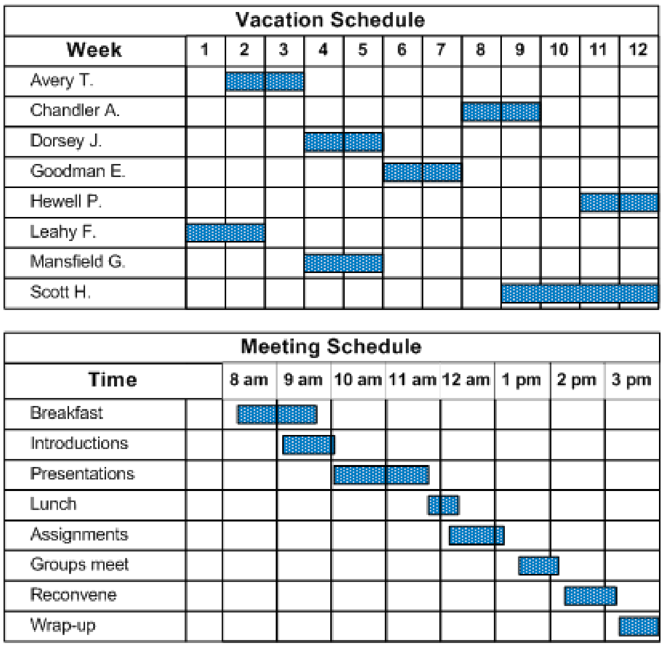
The following figure shows a Gantt chart.



Example of a bar chart used for planning major programs, sometimes referred to as a Gantt chart

Scheduling charts are used to indicate when things will happen, to make sure that there is no overlapping, or to coordinate multiple activities or resources. You could have activities or resources along the vertical axis.

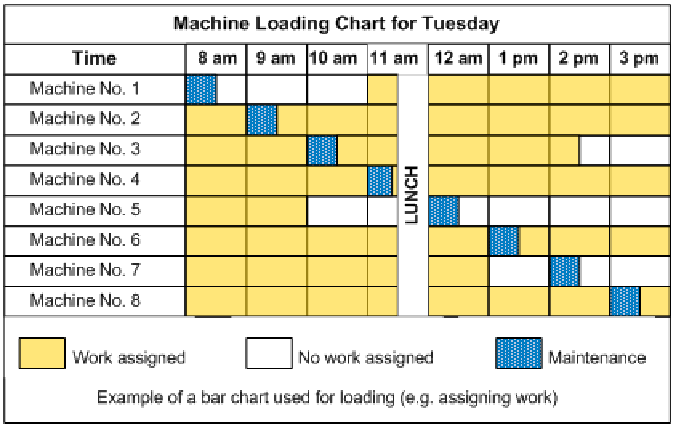
The following figure shows charts for scheduling.



Examples of time and activity bar charts used for scheduling

Charts for showing the work assigned to a resource, such as a person or a machine, represent the loading of that resource. A blank cell indicates that nothing is assigned to a resource in the given period. A filled cell shows that work has been assigned to the resource. The objective is often to achieve maximum utilization of resources, with every cell filled.

The following figure shows a Load chart.



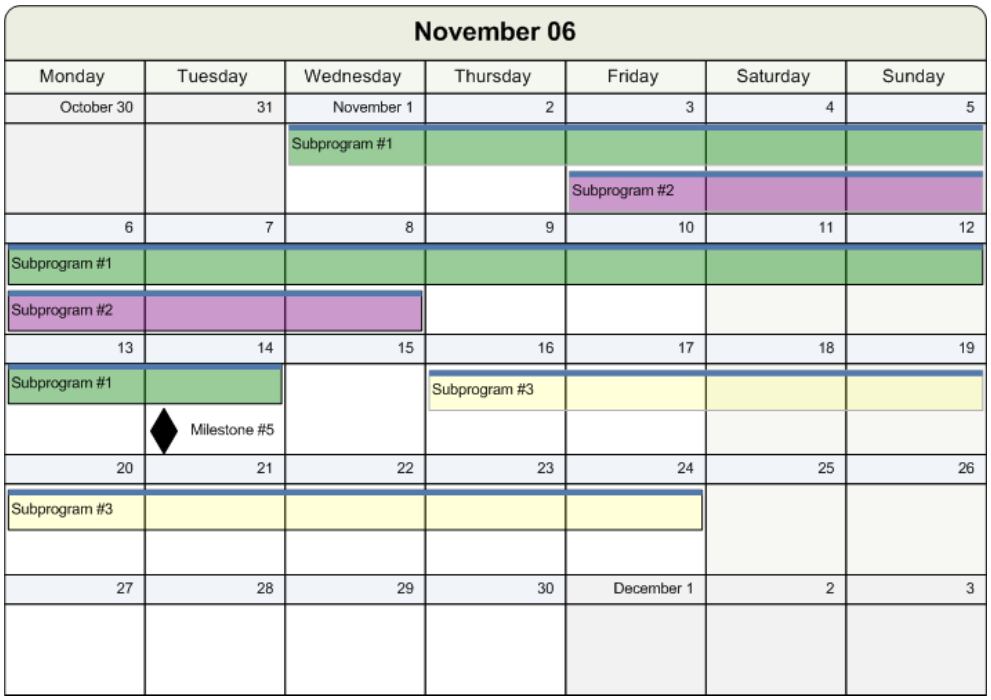
The dependency of an activity upon another can be indicated, usually by an arrowed line pointing from one task to another. For example, if an arrow leads from one bar to another, you can interpret it to mean that the task represented by the bar ending at the base of the arrow must be completed before the task represented by the bar starting at the arrow tip can start.

If bars run without interconnecting arrows, the tasks they represent can be considered to be independent.

A type of chart that displays all interdependencies is called a *PERT chart*.

Activity bar charts can display the relationship of activities to absolute or relative horizontal time scales. In contrast, activity calendar views display the relationship of activities to absolute dates and times arranged in a monthly or hourly grid. This type of chart is useful for showing how activities relate to weekends, holidays, and other milestones that are external to the project schedule.

The following figure shows a monthly calendar view chart.


November 06
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
October 30
31
November 1
2
3
4
5
Subprogram #1
Subprogram #2
6
7
8
9
10
11
12
Subprogram #1
Subprogram #2
13
14
15
16
17
18
19
Subprogram #1
Subprogram #3
Milestone #5
20
21
22
23
24
25
26
Subprogram #3
27
28
29
30
December 1
2
3

The added value of JViews Gantt

JViews Gantt makes it easy to develop charts for planning, scheduling, monitoring, and managing purposes across a wide range of industries. It provides time and activity chart components—two variants of Gantt chart, a Resource Data chart, and two variants of Calendar View—that you can develop and customize to your own requirements. Thus, JViews Gantt can provide different cross-sectional views of the same planning data.

JViews Gantt provides Gantt charts as a task-oriented view, like those found in project-management software, or as a resource-oriented view that shows the scheduling of the activities of a resource. It also offers you a view in the form of a load chart that displays the use of resources along the timeline. Finally, JViews Gantt provides monthly and daily Calendar Views that display activities arranged on monthly or hourly grids.

See *Time and activity charts in JViews Gantt* for examples of the charts that JViews Gantt can offer. These software components can be used as is or developed and customized to suit your own planning and scheduling requirements.

The bars in the Gantt chart show the activities for which the matching resource is reserved.

It can be difficult and costly to develop a robust Gantt charting solution in-house for Java® applications. Off-the-shelf project management tools are not appropriate, because they do not offer all the common views and they are not available in Java.

The ideal solution is JViews Gantt with its ready-to-use set of components for creating Gantt chart displays, its development tools for quickly configuring the look-and-feel without coding, and its comprehensive SDK for providing specialized functionality.

This solution allows deployment in applications, applets, and thin clients. The same code used to configure and populate the chart is reused across deployment platforms.

In particular, JViews Gantt helps you reduce dramatically the time required to develop Web-based scheduling applications. It allows you to build richer, smarter, and more natural user interfaces for this type of application.

When you migrate scheduling applications to the Web, the options available for presenting data effectively can be limited. The application often ends up being delivered with a GUI that is less intuitive than native-platform GUIs, that is, non-Web interfaces.

JViews Gantt is the ideal solution to this problem. It helps you create custom editing and visualization scheduling services that will work over the Web. Specific services address common problem areas for Web GUIs, including platform and browser independence, efficient data transfer, and manipulating large data sets. JViews Gantt helps you obtain intuitive GUIs for Web-based planning by giving you customizable components that precisely address end-user needs.

The data in the Gantt data model can be viewed as a hierarchical table, task-oriented chart, resource-oriented chart, a resource load chart, as a monthly calendar view, and as a daily calendar view. All these views of the data are synchronized.

The task-oriented and resource-oriented Gantt charts and the hierarchical table component can be adapted through the Designer, the product’s own easy-to-use editing GUI. You can develop quickly using productivity tools that include wizards and dialogs to configure Gantt displays without coding. The learning and development time is reduced to a minimum.

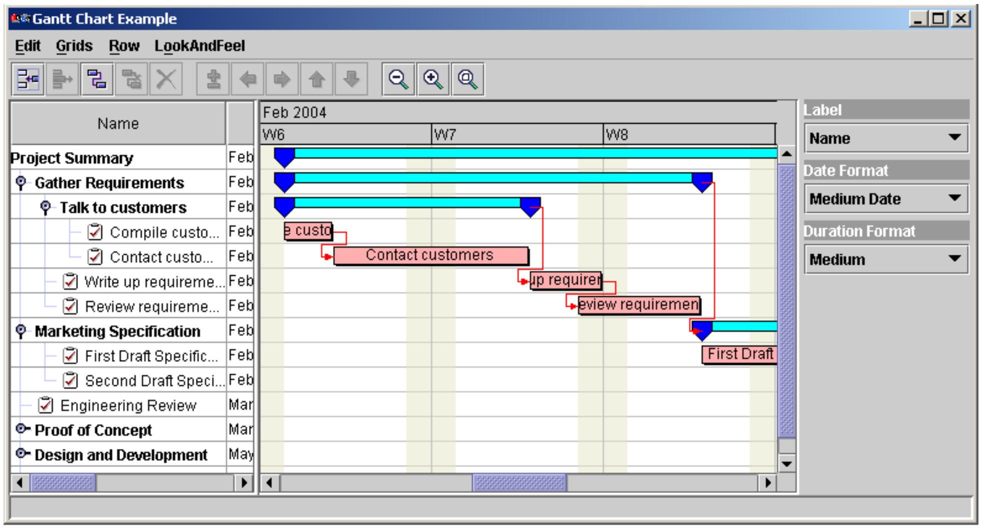
You can still implement the most demanding requirements of an application. You can use the SDK to refine and extend the Gantt chart display. The SDK has a fully documented and supported Java API.

JViews Gantt includes IBM® ILOG® JViews Charts, so you have access to the full range of charting possibilities.

Time and activity charts in JViews Gantt

JViews Gantt provides support for creating, viewing, and editing activity-oriented Gantt charts. This type of Gantt chart shows activities and how they are distributed over time; this is the view you see in most project planning tools, such as Microsoft® Project.

The following figure shows an activity-oriented Gantt chart.

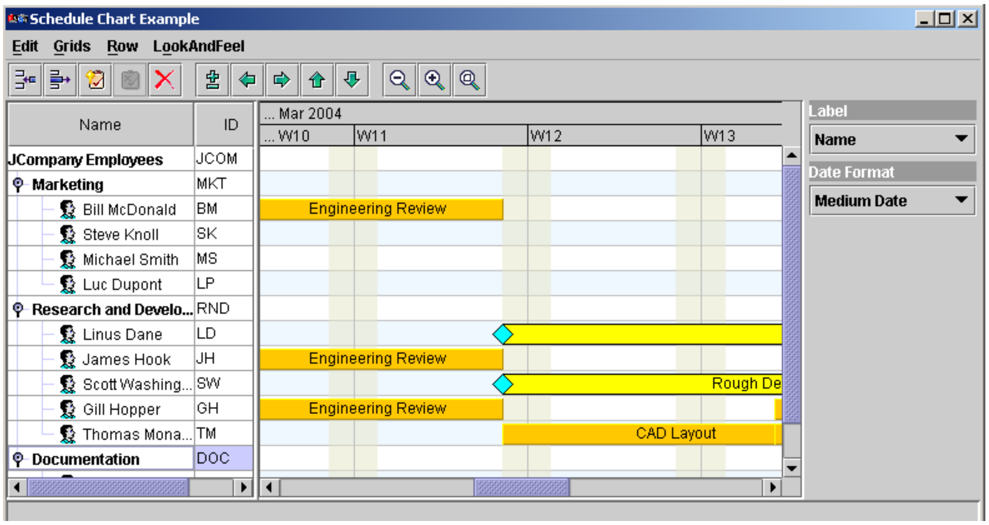


Name	
Project Summary	Feb
Gather Requirements	Feb
Talk to customers	Feb
Compile custo...	Feb
Contact custo...	Feb
Write up requireme...	Feb
Review requireme...	Feb
Marketing Specification	Feb
First Draft Specific...	Feb
Second Draft Speci...	Feb
Engineering Review	Mar
Proof of Concept	Mar
Design and Development	May

The time-dependent relationship between tasks, the constraint, is also typically shown between activities.

JViews Gantt provides support for creating, viewing, and editing resource-oriented Gantt charts. The resource-based Gantt chart, or Schedule chart, shows how resources are scheduled by showing the time periods during which a resource is reserved.

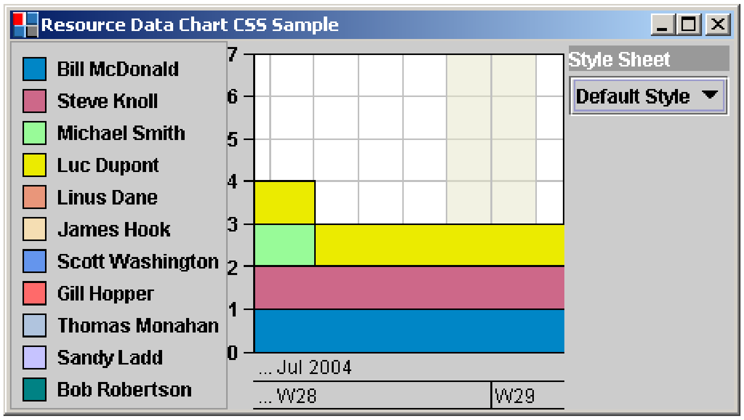
The following figure shows a resource-oriented Schedule chart.



Name	ID
JCompany Employees	JCOM
Marketing	MKT
Bill McDonald	BM
Steve Knoll	SK
Michael Smith	MS
Luc Dupont	LP
Research and Develo...	RND
Linus Dane	LD
James Hook	JH
Scott Washing...	SW
Gill Hopper	GH
Thomas Mona...	TM
Documentation	DOC

JViews Gantt provides support for creating, viewing, and editing data, primarily numerical, on resources and the activities assigned to them. The Resource Data chart is a Cartesian xy chart that displays numerical information related to the activities to which a resource is assigned against time.

The following figure shows a Resource Data chart.



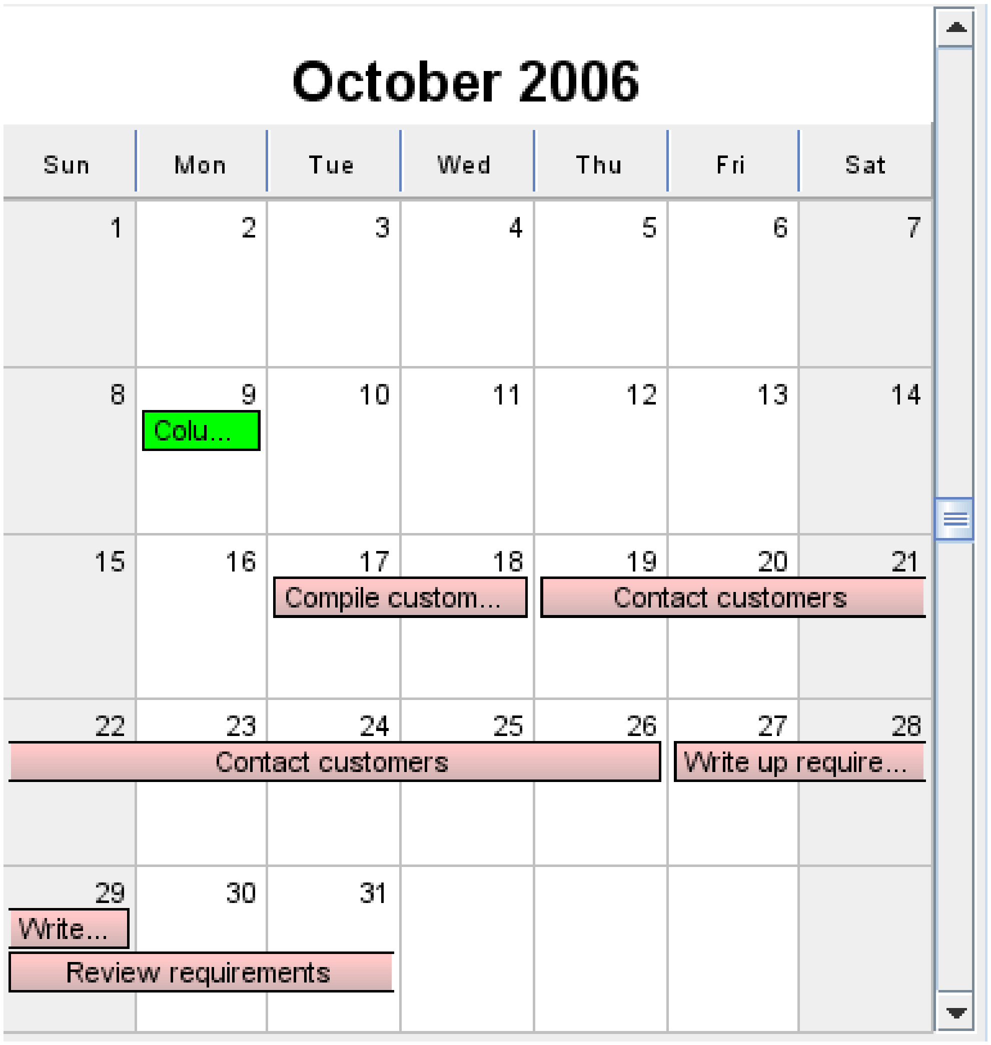
Like the Gantt charts, the horizontal x axis represents time. By default, the Resource Data chart displays the number of activities simultaneously assigned to a resource as the y-axis value. This is called the loading of a resource. The Resource Data chart can be customized easily to display other numerical attributes of activities along the y axis, such as weighted averages, resource availability, resource costs, and complex load calculations.

The Resource Data chart can be rendered in a variety of styles, including polylines, area fills, or bars.

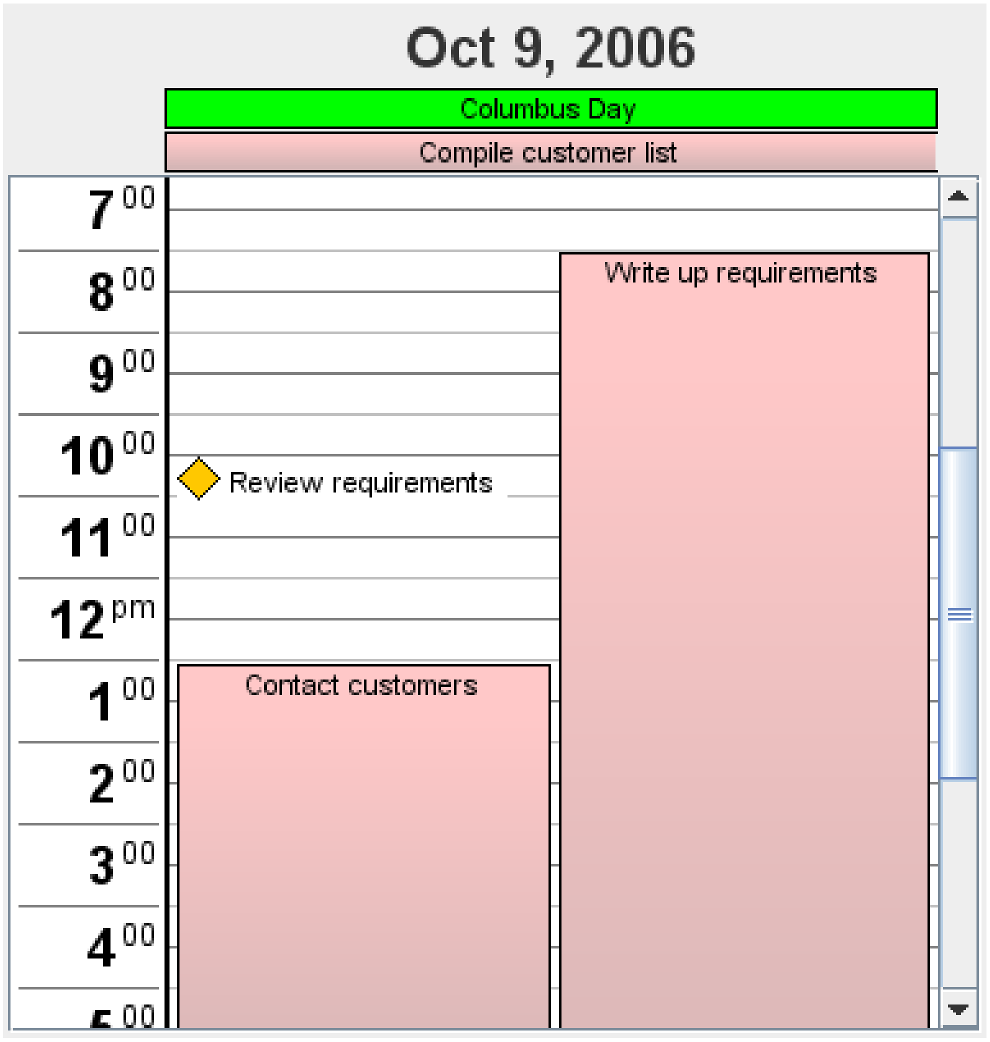
The Resource Data chart can be synchronized easily in several ways with Gantt and Schedule charts that could participate in the same user interface. You can synchronize the horizontal time access of multiple charts and you can synchronize the selection of resources in a Schedule chart with the resource data displayed in a Resource Data chart.

JViews Gantt provides support for creating, viewing, and editing monthly and daily Calendar Views. This type of Gantt view shows activities and how they are distributed over days within a month or hours in a day.

The following figure shows a Gantt chart with activities distributed over months.

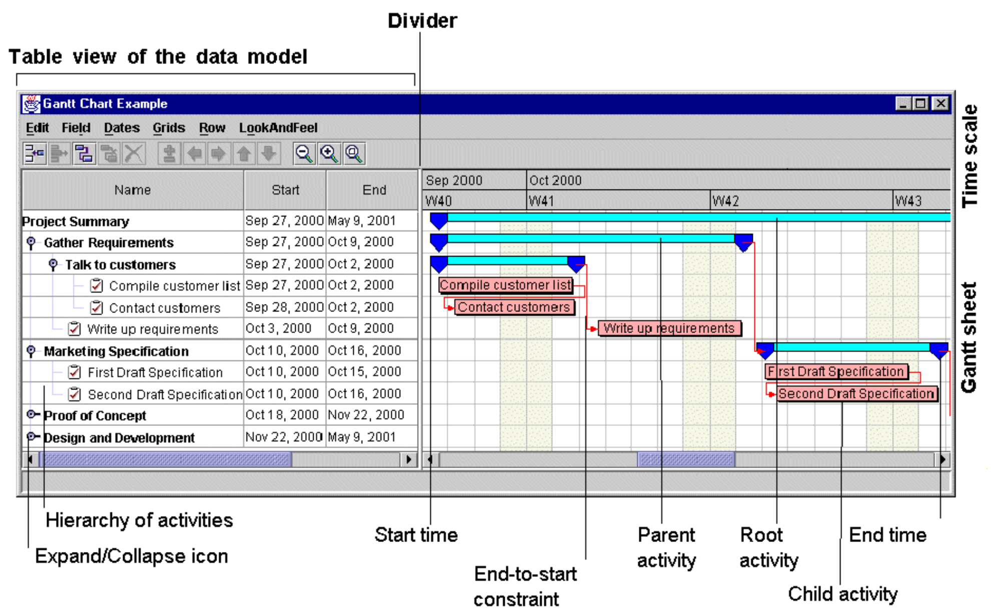


The following figure shows an JViews Gantt calendar view with activities distributed over days.



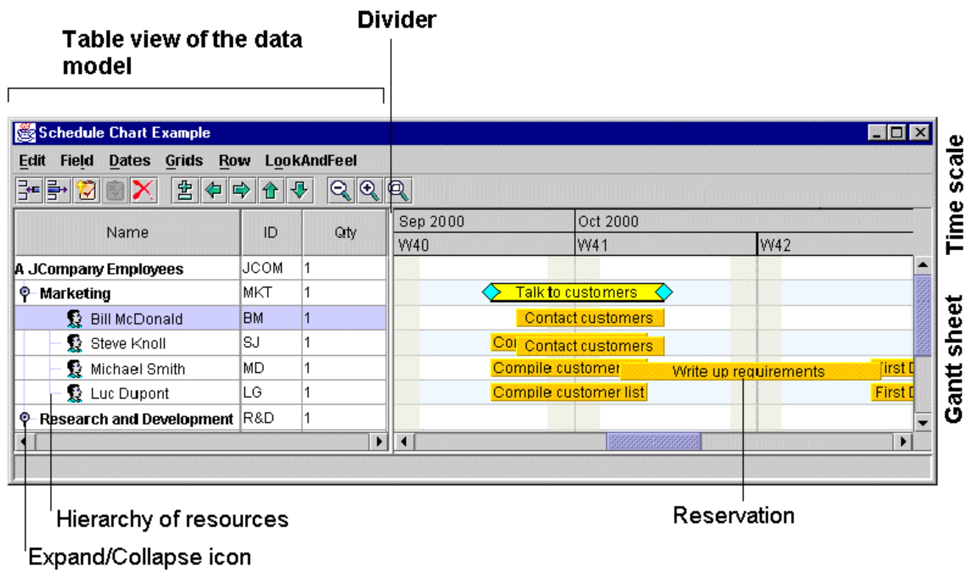
Each chart component is built from smaller components, such as the time scale, the activity table, the activity sheet, the resource table, and the resource sheet. You can connect the chart components to your business data and display various representations of the scheduling data. You can arrange the user interface components in almost any layout to display this data.

The following figure illustrates the Gantt chart components.



Name	Start	End
Project Summary	Sep 27, 2000	May 9, 2001
Gather Requirements	Sep 27, 2000	Oct 9, 2000
Talk to customers	Sep 27, 2000	Oct 2, 2000
Compile customer list	Sep 27, 2000	Oct 2, 2000
Contact customers	Sep 28, 2000	Oct 2, 2000
Write up requirements	Oct 3, 2000	Oct 9, 2000
Marketing Specification	Oct 10, 2000	Oct 16, 2000
First Draft Specification	Oct 10, 2000	Oct 15, 2000
Second Draft Specification	Oct 10, 2000	Oct 16, 2000
Proof of Concept	Oct 18, 2000	Nov 22, 2000
Design and Development	Nov 22, 2000	May 9, 2001

The following figure shows the schedule chart components.



Name	ID	Qty
A JCompany Employees	JCOM	1
Marketing	MKT	1
Bill McDonald	BM	1
Steve Knoll	SJ	1
Michael Smith	MD	1
Luc Dupont	LG	1
Research and Development	R&D	1

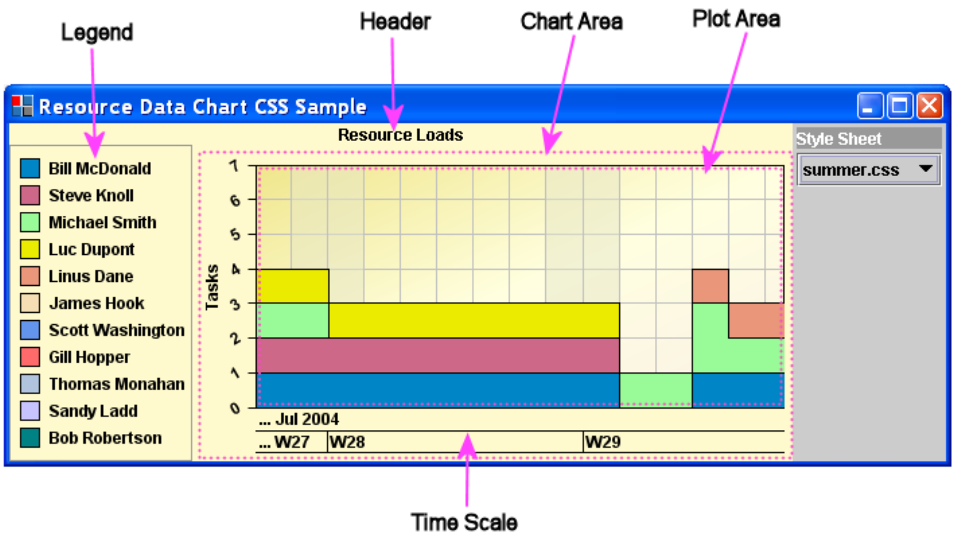
In the industry, what is referred to as a Gantt chart normally corresponds to the activity sheet or the resource sheet of a Gantt chart in JViews Gantt; that is, a generic Gantt chart corresponds to the graphic part of a Gantt chart in JViews Gantt. JViews Gantt provides different views of the information held in these sheets, so you can also have a table view or a tree view.

The table in the Gantt charts is derived from the Swing class `JTable`. Any customization allowed on a `JTable` object is also possible on the Gantt table. The table view can be hidden or visible. The previous two figures show examples of the table view.

The tree is a column in the table that shows a hierarchical structure of activities or resources. By default, the tree is the first column in the table. The tree is derived from the Swing `JTree` class adapted to be a column of a table.

The rows in the table are activities or resources. The tree column displays properties of activities or resources. By default, the name property of the given property or resource is displayed.

The following figure shows the Resource Data chart component and its subcomponents.



Main features of JViews Gantt

Describes the main features of JViews Gantt.

In this section

Performance and Scalability
Explains how JViews Gantt supplies load-on-demand as an efficient way of handling large data sets in custom data models.

Fully featured printing
Describes the printing features and its benefits.

Flexible deployment
Explains how a JViews Gantt application can be deployed on a client as an applet or application and on a server to generate DHTML pages.

Overview

JViews Gantt comes ready-to-use and ready-to-customize. It offers you the following features:

- ♦ Gantt, Schedule, and Resource Data charts, monthly and daily Calendar Views, plus use with IBM® ILOG® JViews Charts
- ♦ Rapid application development through the GUI of the Designer
- ♦ MVC architecture
- ♦ JDBC and XML data connections
- ♦ Easy customization
- ♦ Easy styling
- ♦ Complete set of interactions
- ♦ Optimized performance and scalability for handling large data sets
- ♦ Fully featured printing
- ♦ Use with IBM® ILOG® JViews Diagrammer for PERT representations
- ♦ Use with IBM® ILOG® JViews Maps, for example, in dispatching applications
- ♦ Deployment as Java Bean®
- ♦ Deployable on client or server
- ♦ Advanced capabilities for thin-client applications leveraging Asynchronous JavaScript® And XML (Ajax) concepts with JavaServer Faces (JSF) as the server-side component model and Dynamic HTML (DHTML) as the client-side display technology

Chart components

JViews Gantt comes with three chart components and two calendar view components. The Gantt charts are either activity-based (Gantt Chart) or resource-based (Schedule chart). There is also a Resource Data chart for presenting numerical information about resources and the activities assigned to them.

JViews Gantt provides multiple views of data. For example, within a chart you have the Gantt table and the Gantt sheet view of the data. See *Time and activity charts in JViews Gantt* for an illustration. The table view displays the resources or the activities with their properties. Such a view is hierarchical; for example, an activity can contain subactivities and can be collapsed or expanded.

Task-oriented views are typically used in project management applications. Each task is associated with a row in the Gantt chart or a bar in the Calendar Views.

Resource-oriented views are typically used in manufacturing and transportation applications. Each row of the Gantt (or Schedule) chart is associated with a resource and the view displays in a row all the activities assigned to the resource.

The Resource Load view displays the workload of a specific resource in time.

When you purchase IBM® ILOG® JViews Gantt, you get IBM® ILOG® JViews Charts as well for general purpose charting. You can customize the Resource Load chart. You can integrate applications developed with JViews Gantt with JViews Charts; for example, you can display trends and indicators by means of bar charts, pie charts, radar charts, and real-time charts. Full documentation is supplied in the JViews Charts package.

It is also possible to develop a PERT representation with JViews Diagrammer and to integrate this chart with JViews Gantt.

The Gantt Viewer sample found at **`<installdir>/jviews-gantt86/samples/ganttViewer/src/ganttviewer/GanttViewer.java`** shows how to integrate a PERT chart with different types of Gantt chart, a Resource Data chart, and the monthly Calendar view, if you have JViews Diagrammer installed. See *Complete planning application with PERT chart*.

JViews Gantt provides an open and wide-ranging set of features with the following benefits:

♦ Specific types of Gantt chart and a load chart adapted to specific application domains.

♦ Integration with other types of chart included in the product.

♦ Use with specific features of other IBM® ILOG® visualization products, such as PERT representations and deployment maps.

Rapid application development

JViews Gantt is ready-to-use. Each chart component is supplied as a Java Bean® with default styles and a full set of interactions. You can create, move, and resize activities, select, add constraints between tasks, pan, zoom with smart time scale, and access tool tips. JViews Gantt offers you best-in-class interactions.

All these capabilities are available through an easy-to-use GUI, the Designer. It is easy to interact with this tool to develop your Gantt chart components.

JViews Gantt lets you develop the charts completely in the GUI. A chart component can even be integrated as a JavaBean into a third-party IDE, so that you can continue to develop within your preferred environment, but using the appropriate parts of the Designer to develop the chart components.

If you can use JDBC to connect your database, if your data is in XML, if you can export your data to a flat file (CSV) format, or if you want to use an in-memory data model for testing, you can start to develop in the GUI. In this case, the classes of the library used to display the scheduling data are hidden from you. You can develop your application through a normal windowing interface with the usual interaction features, such as wizards and point-and-click. You choose the chart component that best suits your needs and customize the styling properties. There is no need to write any Java® code at this stage or to be familiar with Cascading Style Sheets (CSS) syntax.

The Designer includes the unique Style Rules feature, which allows you to define how the graphics will appear on the screen under different data conditions. Using the Designer to style your component hides the complexity of CSS. It is a very efficient way of building a chart component, which cuts development time considerably.

The output of the Designer is a project file that is loaded into the application at run time.

In fact, if you develop entirely through the Designer, the only time you will need to write any Java code is when you integrate a chart component into an application.

For more about the developing with the Designer, see *Creating a Gantt chart with the Designer*.

JViews Gantt gives you:

♦ Tool tips

♦ Weekend grid

♦ Overlapping task layout options

♦ Smart time scales

♦ Tasks and summary tasks

The benefits are:

♦ You can integrate your own data easily through JDBC, XML, flat file, or in-memory.

♦ Development time is cut because of:

  - Ready-made Java Bean® components with many included features.
  - Default styles and interactors.

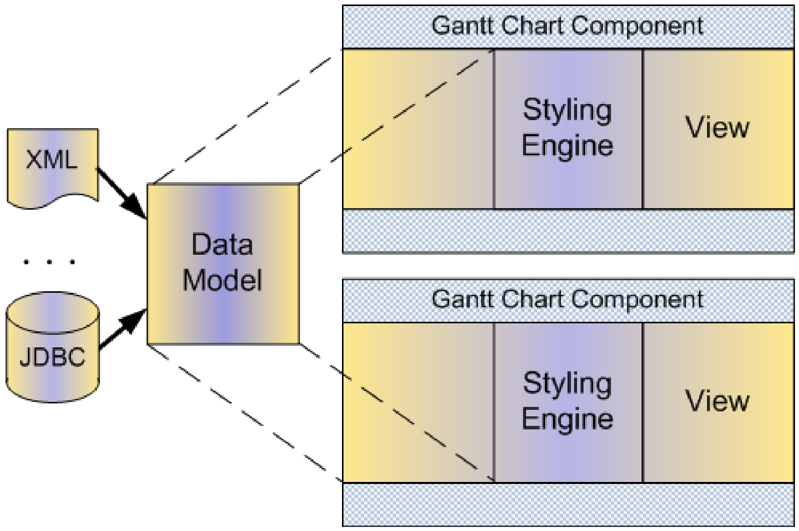
- Easy-to-use GUI.
- No need to use Java or CSS explicitly to develop components.

MVC architecture

JViews Gantt makes use of a powerful Swing-like MVC architecture. MVC architecture means that there is a clear separation between data and display, consistent with Swing Java® user interface. The data model is open and extensible, as opposed to other types of closed architecture.

MVC architecture allows for multiple views from the same data model.

The following figure shows the data model with multiple views and data connection.



The model and the view use a publish/subscribe mechanism to notify each other about changes in the model or the view. For example, the model will publish information about updates to the model; the view subscribes to listen for any modifications to the model. The same mechanism operates in the opposite direction, so that the model subscribes to listen for updates published by the view. The scheduling data model acts as a consistency manager across views and communicates with the rest of the application.

JViews Gantt can be used as is to represent activities and subactivities, resources, constraints, and reservations.

The benefits are:

♦ Familiar MVC concept

♦ Open Architecture

♦ Multiple views from the same data model

Data connection

JViews Gantt comes with JDBC, XML, flat file, and in-memory data connections ready to configure, but it can connect to any data source. The open API gives you flexibility to integrate with any back end.

XML support is through the Schedule Data eXchange Language (SDXL) for serializing data. SDXL allows the easy interchange of scheduling data.

If you need to connect to a custom data model, you must integrate it through the Java® API of the SDK instead of using the Designer.

The benefits are:

- JViews Gantt is open to any back-end data connection by use of the API.
- No API development necessary for JDBC database connection, XML, flat file, or in-memory data model.

Easy customization

JViews Gantt is ready-to-customize. It is easy to create custom renderers and customizable interactors, either through the Designer or the fully accessible API for additional graphic refinement.

The table and the tree are customizable. See *The added value of JViews Gantt* for details about the table and the tree.

You can style screen objects by creating new styles easily.

The benefits are:

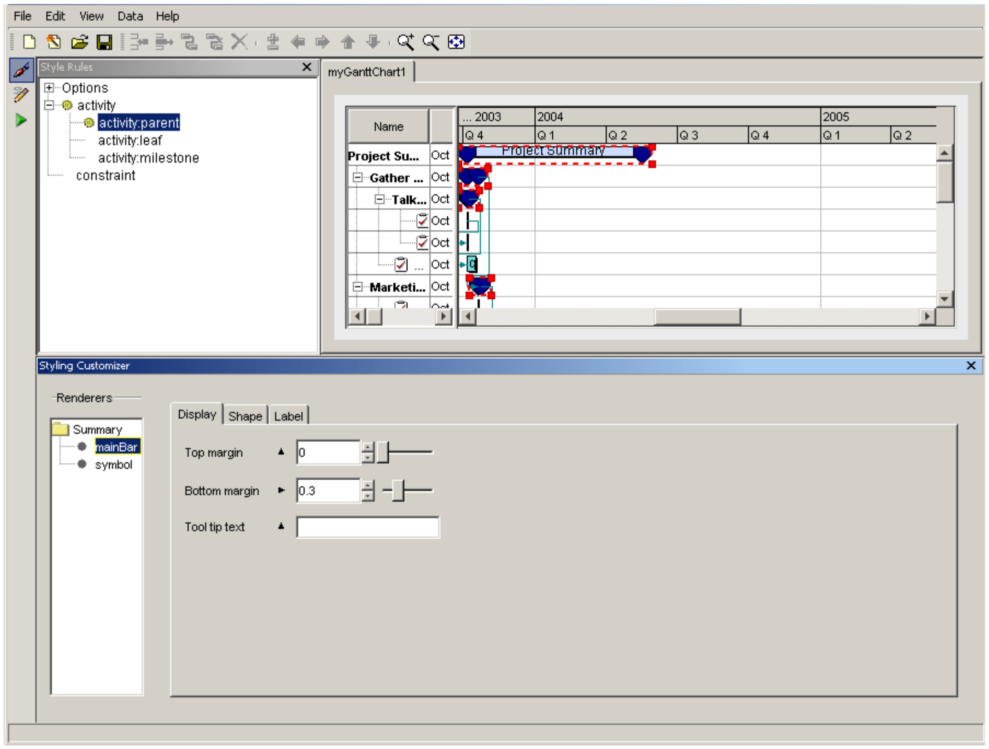
♦ Large range of customization available with minimum effort in the Designer.

♦ Styling done in the Designer can be fully integrated with the Java® API, if you need to do further graphic refinements.

Easy styling

Styling in JViews Gantt is based on CSS and on a specific extension of CSS, CSS for Java®. You do not need to know CSS syntax in detail. The Designer hides the complexity of this syntax and lets you develop your styles by applying style rules through an editor based on natural language. Specific customizers allow you to refine styling with ease through the GUI. For advanced styling refinements, you can continue to develop your styling in the API.

The following image shows the customizer in the Designer.



Top margin 0

Bottom margin 0.3

Tool tip text

You can write style rules to apply styling selectively, depending on the data conditions.

The benefits are:

♦ Easy to achieve sophisticated styling and rendering.

♦ No need to write explicit CSS syntax.

Complete set of interactions

JViews Gantt provides a full set of interactions.

In the activity-based and resource-based Gantt charts, the following Gantt sheet interactions are installed by default:

- ♦ Zoom-in and zoom-out
- ♦ Horizontal time scrolling and vertical scrolling
- ♦ Selecting, moving, and resizing activity graphics
- ♦ Tool tip

The following additional Gantt sheet interactions are available in Chart Editing Mode or with the SDK:

- ♦ Zoom to fit
- ♦ Making activities or resources
- ♦ Making constraints or reservations

The following time scale interactions are installed by default:

- ♦ Drag for scrolling horizontal time
- ♦ Click or double-click for zooming in or out

The following additional interactions are provided in the Designer and the samples:

- ♦ Deleting or inserting activities, resources, reservations, or constraints
- ♦ Expanding or collapsing rows
- ♦ Outline interactions: indent, outdent, up, or down

In Resource Data charts, the following interactions are installed by default:

- ♦ Selection
- ♦ Tool tip
- ♦ Legend docking

In the Resource Data chart, all the interactions provided in the JViews Charts SDK are available.

The benefits are:

- ♦ The large range of default interactions gives you ready-made interactors to cover most of your requirements without writing any code.
- ♦ Special cases can be met by using the SDK; in some cases, the samples supply the required code ready-made.
- ♦ Access to Charts interactions for Resource Data charts.

Performance and Scalability

The Gantt chart is optimized to allow fast refreshes and interactions. Schedule data often includes several thousands of activities. Very large data sets are handled seamlessly with a load-on-demand mechanism that fetches the data when it becomes visible.

JViews Gantt provides load-on-demand as an efficient way of handling large data sets in custom data models with significant time latency in retrieving activities assigned to resources. Manipulating large amounts of data in one set can slow down response times. Therefore, JViews Gantt lets you load data into memory as it is needed for display in a chart, so that you load only the relevant part of the data and performance is not adversely affected. There is no lengthy delay as the data is loaded.

Both types of Gantt chart automatically apply vertical load-on-demand with both default and custom data models. Vertical load-on-demand is the ability to defer loading row-oriented information until it is needed for display. Row-oriented information is activity data for the Gantt chart and resource data for the Schedule chart. Vertical load-on-demand applies to visible rows only. A visible row is a row that has all its parent rows expanded. A visible row is not the same as a displayed row. A displayed row is both visible and scrolled into the display window. Vertical load-on-demand does not operate on displayed rows. For example, if 20 activities are visible, but only 10 are displayed, all 20 activities are loaded. If only first level activities in the outline are visible and none of them are expanded to make their child activities visible, only the first level nodes are loaded.

The principle of vertical load-on-demand is applied in the Resource Data chart too. In the Resource Data chart, you can only query the data model for the resources that you are currently charting.

Horizontal load-on-demand is available with Schedule charts that are fed by custom data models that can query back-end data selectively. Horizontal load-on-demand is the ability to defer loading reservation data based on the currently visible time interval. As the chart is scrolled or zoomed horizontally to display different time intervals, the Schedule chart queries the data model selectively for new reservations to display.

The default data model is not susceptible to optimization of this kind. It is not able to query back-end data for a time period. It returns all reservations for all time periods and then performs the filtering at the level of the chart.

The JDBC connection applies selective querying of certain properties at the database level, thus leveraging horizontal load-on-demand.

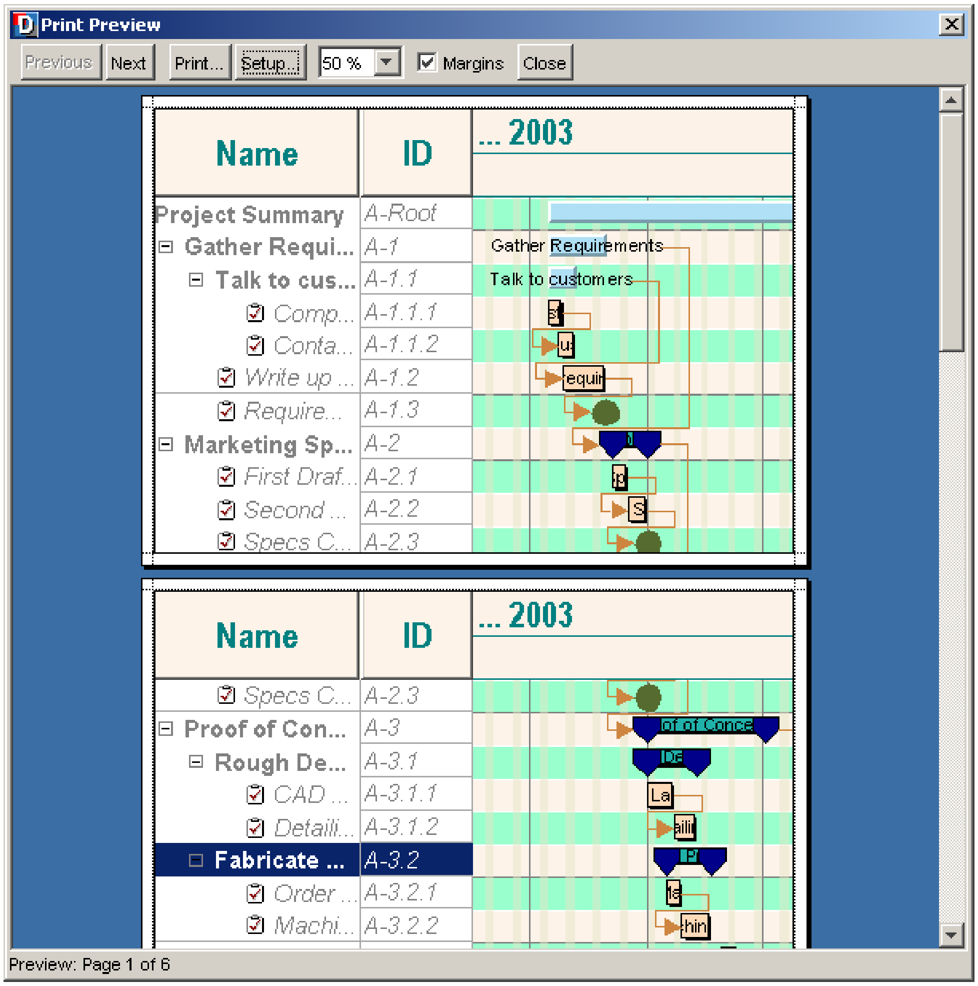
The benefits are:

- ♦ Better performance
- ♦ Scalable applications

Fully featured printing

JViews Gantt has a full printing API. It offers multipage printing and print preview. The printing capability is also available from the Designer.

The following figure shows the print preview of a multipage document in JViews Gantt.



50 % · ☑ Margins

Name	ID	... 2003
Project Summary	A-Root	
Gather Requi...	A-1	
Talk to cus...	A-1.1	
Comp...	A-1.1.1	
Conta...	A-1.1.2	
Write up ...	A-1.2	
Require...	A-1.3	
Marketing Sp...	A-2	
First Draf...	A-2.1	
Second ...	A-2.2	
Specs C...	A-2.3	

Name	ID	... 2003
Specs C...	A-2.3	
Proof of Con...	A-3	
Rough De...	A-3.1	
CAD ...	A-3.1.1	
Detaili...	A-3.1.2	
Fabricate ...	A-3.2	
Order ...	A-3.2.1	
Machi...	A-3.2.2	

Preview: Page 1 of 6

The benefits are:

- ♦ The ability to achieve sophisticated printing results
- ♦ Easily achievable through the Designer

Flexible deployment

JViews Gantt can be deployed on a client as an applet or application. It can be deployed on the server to generate DHTML pages.

Going beyond simple thin clients, JViews Gantt thin client leverages the local execution capabilities of JavaScript® to provide an advanced user experience; for demanding interactions, Asynchronous JavaScript And XML concepts, or Ajax, are applied.

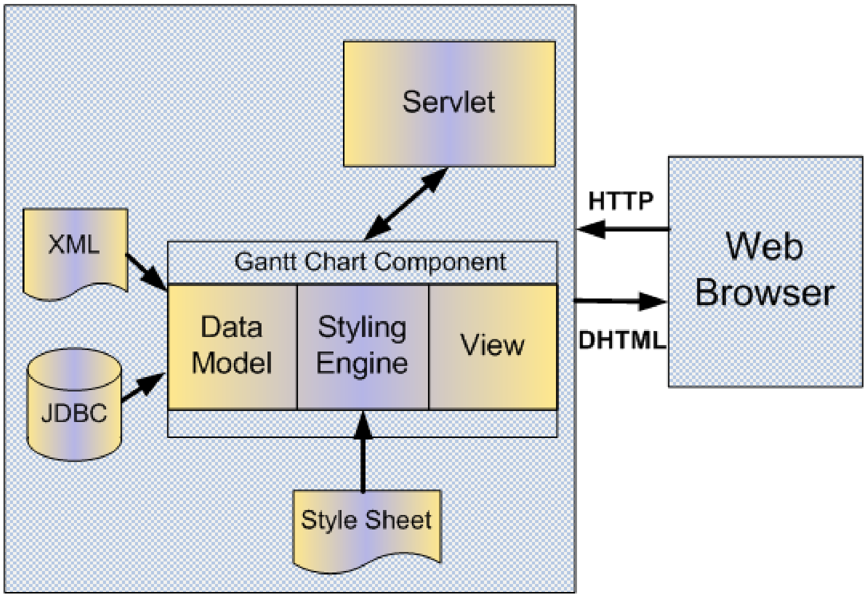
With JViews Gantt Faces components and JavaScript you can develop a new generation of highly responsive, highly interactive Web applications. The high responsiveness is achievable through Ajax, which supports asynchronous and partial refreshes of a Web page.

The client can continue processing while the server processes in the background. A user can continue interacting with the client without noticing latency in the response. The client does not have to wait for a response from the server before continuing, as in the traditional synchronous approach.

JViews Gantt components can be deployed as JavaBeans.

The thin client deployment has the same visually rich display as the Java® client: tree, table, Gantt, and load chart views. The rich and thin client displays are consistent.

The following figure shows a client and Server deployment.



You can also deploy your application with a map, by purchasing and integrating with JViews Maps. This facility is useful for applications such as dispatching systems.

The benefits are:

♦ A large range of deployment possibilities

- Consistent behavior between rich and thin clients

Typical Uses of JViews Gantt

Describes typical uses of JViews Gantt.

In this section

Overview

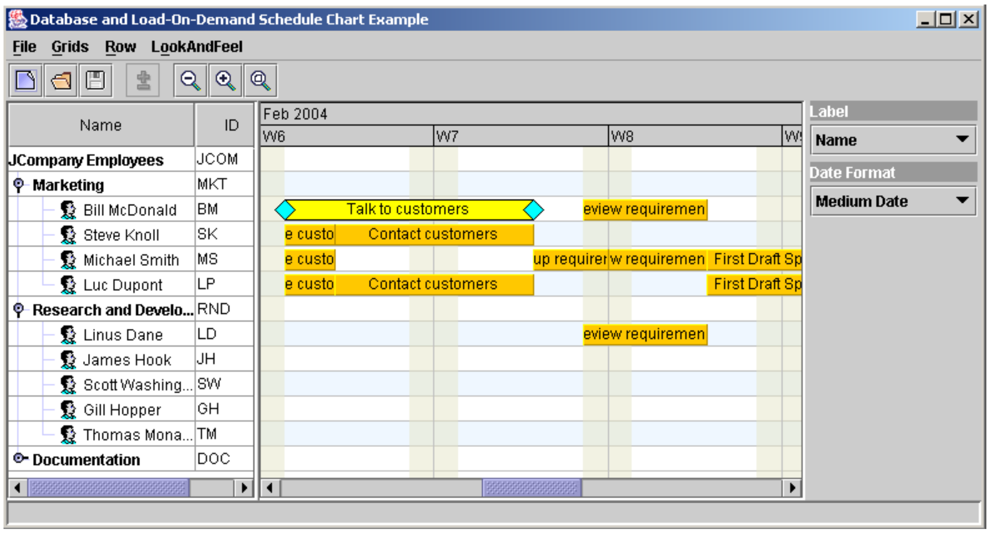
You can use JViews Gantt for:

- Project Planning, including access to Microsoft® Project data if required
- Web-based project plan viewers
- Factory-flow machine scheduling
- Visualization of IBM® ILOG® Scheduler data
- Airline resource allocation
- Resource load usage
- Complete planning application with a PERT representation developed with JViews Diagrammer
- Vehicle-dispatching, with a map showing the routes of trucks (use with JViews Maps)

  An example of this type of application is the integration with IBM® ILOG® Dispatcher.

Project planning

The following figure shows the Schedule chart used in a typical project planning application.



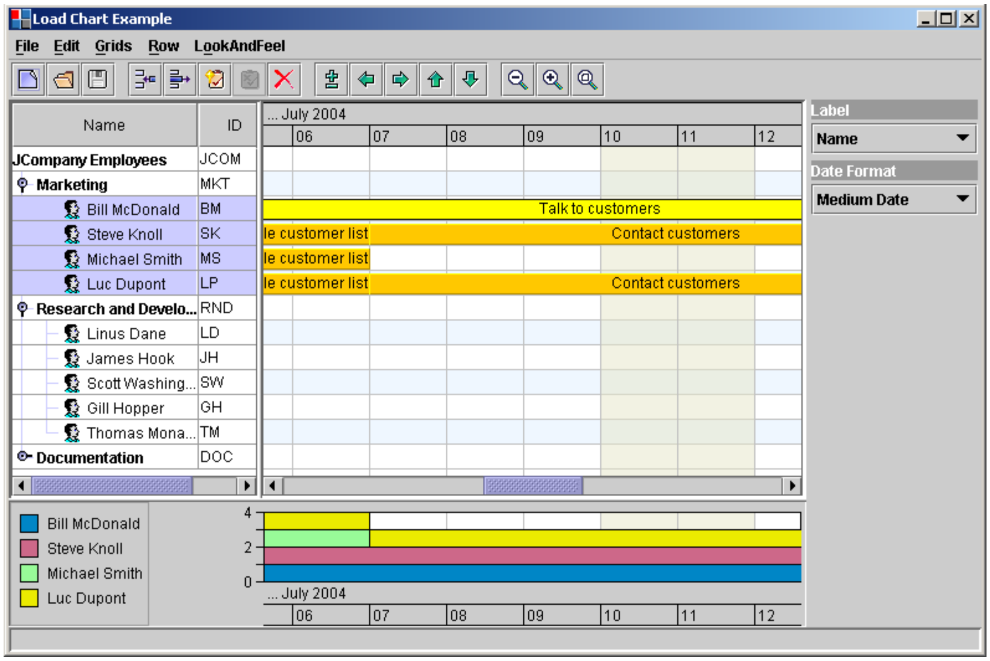
Name	ID
JCompany Employees	JCOM
Marketing	MKT
Bill McDonald	BM
Steve Knoll	SK
Michael Smith	MS
Luc Dupont	LP
Research and Develo...	RND
Linus Dane	LD
James Hook	JH
Scott Washing...	SW
Gill Hopper	GH
Thomas Mona...	TM
Documentation	DOC

Label: Name

Date Format: Medium Date

Resource loading

The following figure shows the Resource Data chart inset below the Schedule chart and used to show numerical data about the loading of resources.

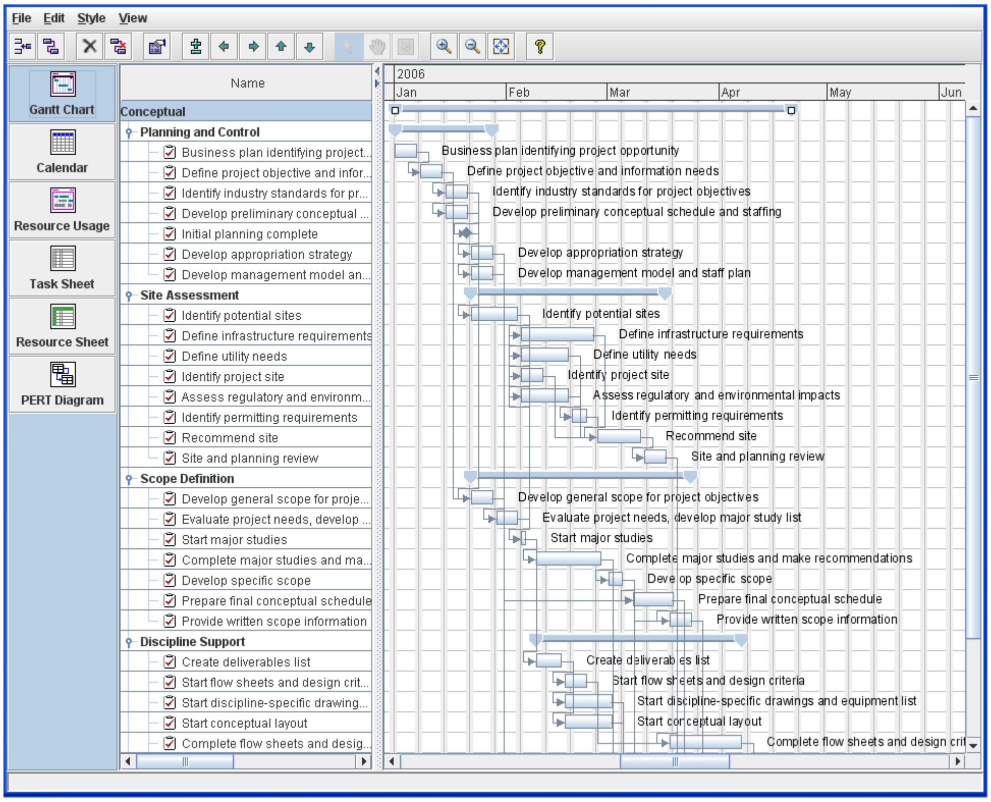


Complete planning application with PERT chart

The Gantt Viewer sample found at **`<installdir>/jviews-gantt86/samples/ganttViewer/src/ganttviewer/GanttViewer.java`** demonstrates a complete planning application that includes an activity-based Gantt chart, a resource-based Gantt chart integrated with a Resource Data chart to show the loading of resources, a monthly Calendar view, a task sheet, and a resource sheet.

This sample has a PERT view that is only visible if IBM® ILOG®JViews Diagrammer is also installed (as a purchased product or as an evaluation). If JViews Diagrammer is installed, the sample will show a PERT chart diagram view of the data in the Gantt data model.

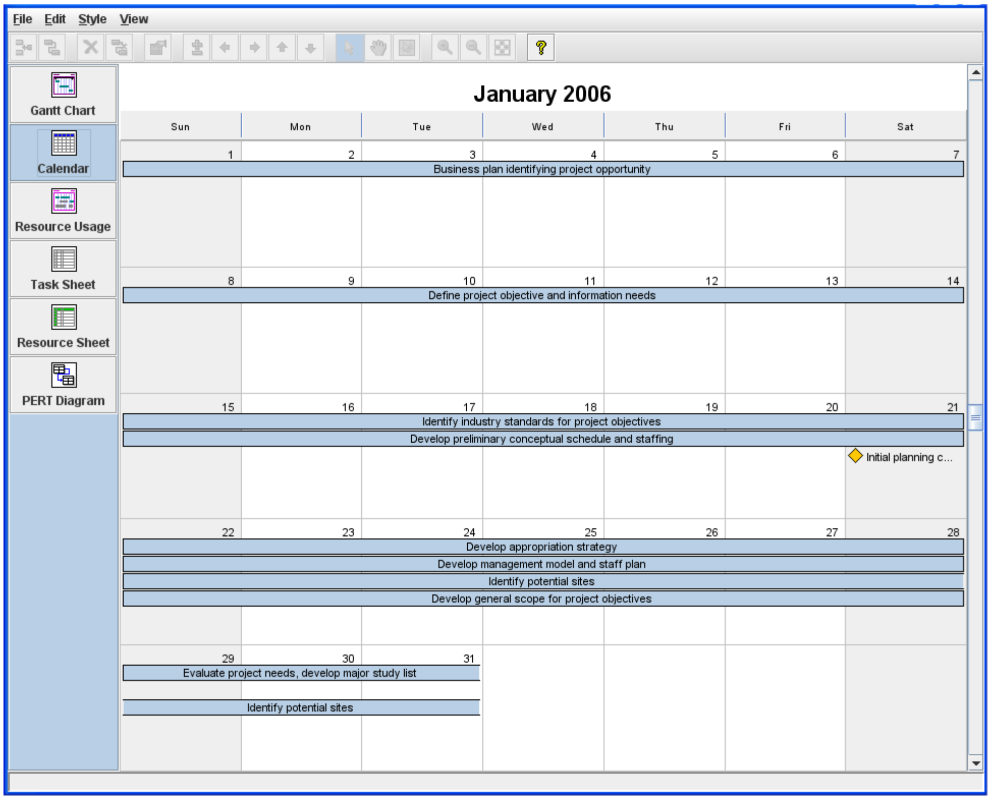
The following figure shows an activity-based Gantt chart from the Gantt Viewer sample.



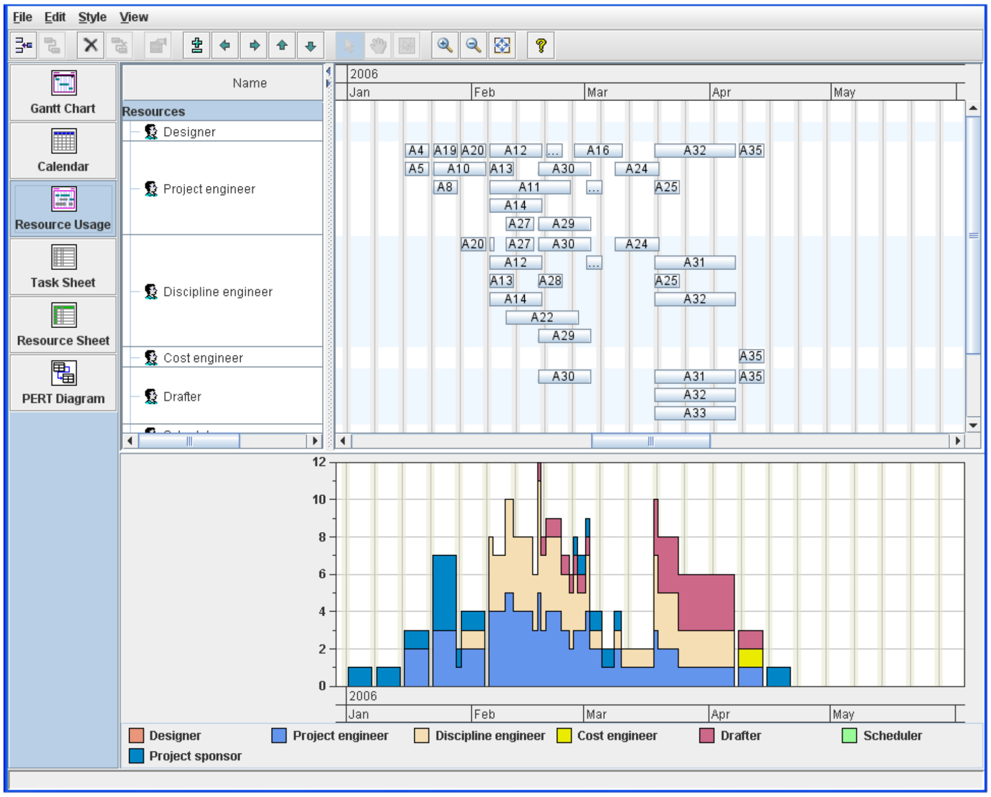
Name
Conceptual
Planning and Control
Business plan identifying project...
Define project objective and infor...
Identify industry standards for pr...
Develop preliminary conceptual ...
Initial planning complete
Develop appropriation strategy
Develop management model an...
Site Assessment
Identify potential sites
Define infrastructure requirements
Define utility needs
Identify project site
Assess regulatory and environm...
Identify permitting requirements
Recommend site
Site and planning review
Scope Definition
Develop general scope for proje...
Evaluate project needs, develop ...
Start major studies
Complete major studies and ma...
Develop specific scope
Prepare final conceptual schedule
Provide written scope information
Discipline Support
Create deliverables list
Start flow sheets and design crit...
Start discipline-specific drawing...
Start conceptual layout
Complete flow sheets and desig...

Business plan identifying project opportunity
Define project objective and information needs
Identify industry standards for project objectives
Develop preliminary conceptual schedule and staffing
Develop appropriation strategy
Develop management model and staff plan
Identify potential sites
Define infrastructure requirements
Define utility needs
Identify project site
Assess regulatory and environmental impacts
Identify permitting requirements
Recommend site
Site and planning review
Develop general scope for project objectives
Evaluate project needs, develop major study list
Start major studies
Complete major studies and make recommendations
Develop specific scope
Prepare final conceptual schedule
Provide written scope information
Create deliverables list
Start flow sheets and design criteria
Start discipline-specific drawings and equipment list
Start conceptual layout
Complete flow sheets and design crit

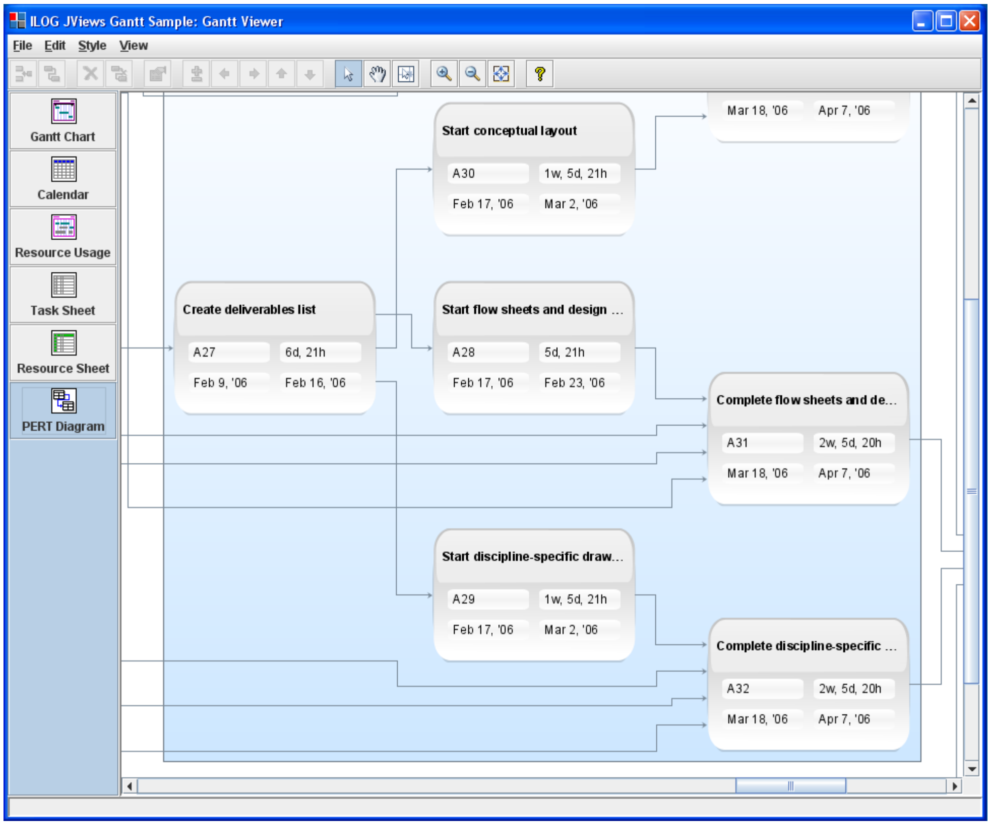
The following figure shows the calendar view from the Gantt Viewer sample.


File Edit Style View
Gantt Chart
Calendar
Resource Usage
Task Sheet
Resource Sheet
PERT Diagram
January 2006
Sun Mon Tue Wed Thu Fri Sat
Business plan identifying project opportunity
Define project objective and information needs
Identify industry standards for project objectives
Develop preliminary conceptual schedule and staffing
Initial planning c...
Develop appropriation strategy
Develop management model and staff plan
Identify potential sites
Develop general scope for project objectives
Evaluate project needs, develop major study list
Identify potential sites

The following figure shows a resource-based Gantt chart with Resource Data chart from the Gantt Viewer sample.

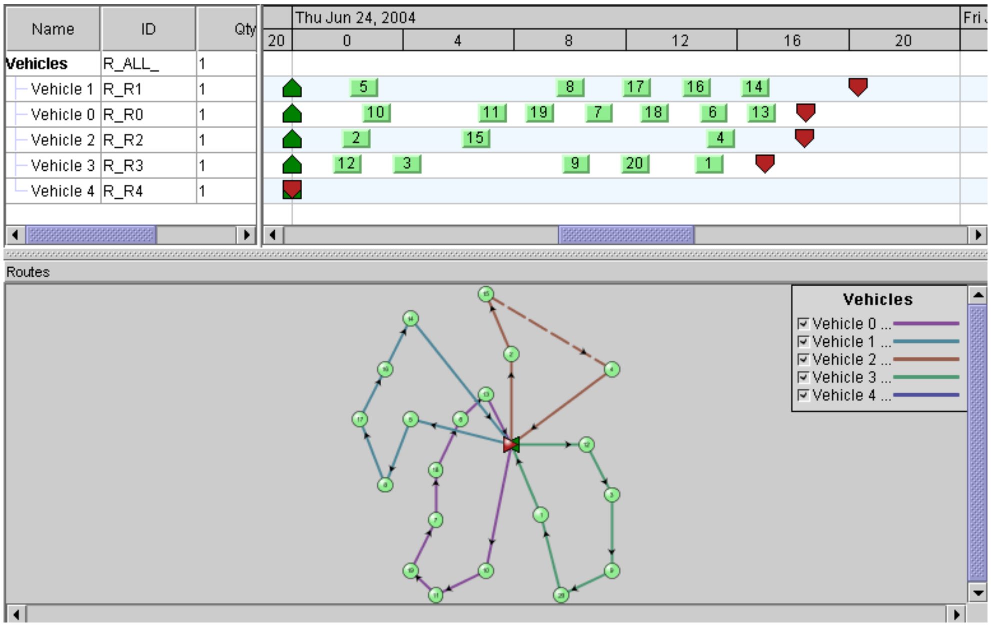


The following figure shows the PERT chart available (if IBM® ILOG® JViews Diagrammer is also installed) in the sample.


ILOG JViews Gantt Sample: Gantt Viewer
File
Edit
Style
View
Gantt Chart
Calendar
Resource Usage
Task Sheet
Resource Sheet
PERT Diagram
Mar 18, '06
Apr 7, '06
Start conceptual layout
A30
1w, 5d, 21h
Feb 17, '06
Mar 2, '06
Create deliverables list
A27
6d, 21h
Feb 9, '06
Feb 16, '06
Start flow sheets and design ...
A28
5d, 21h
Feb 17, '06
Feb 23, '06
Complete flow sheets and de...
A31
2w, 5d, 20h
Mar 18, '06
Apr 7, '06
Start discipline-specific draw...
A29
1w, 5d, 21h
Feb 17, '06
Mar 2, '06
Complete discipline-specific ...
A32
2w, 5d, 20h
Mar 18, '06
Apr 7, '06

Vehicle Dispatching

The following figure shows the use of JViews Gantt with JViews Maps to visualize a vehicle dispatching system implemented in IBM® ILOG® Dispatcher.



Name	ID	Qty
Vehicles	R_ALL_	1
Vehicle 1	R_R1	1
Vehicle 0	R_R0	1
Vehicle 2	R_R2	1
Vehicle 3	R_R3	1
Vehicle 4	R_R4	1

Basic concepts

Presents the basic concepts to help you evaluate JViews Gantt and appreciate its features and capabilities.

In this section

Data

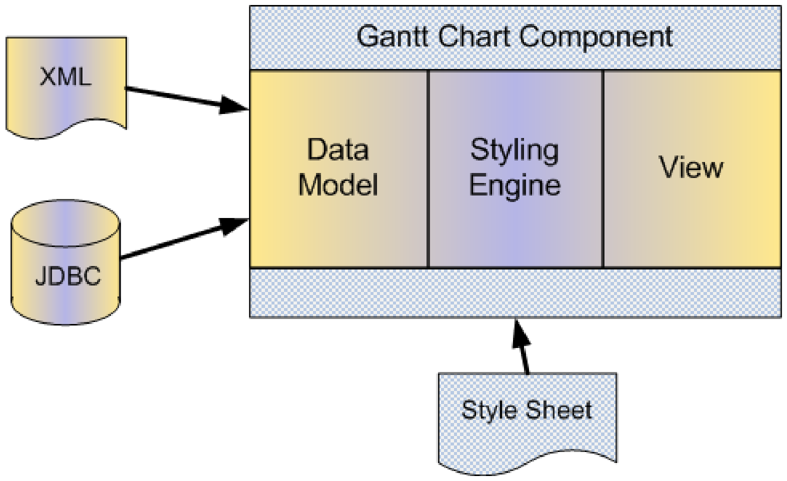
Describes the JViews Gantt data model.

In this section

Overview

JViews Gantt is based on a model-view architecture that cleanly separates the data, the display, and the interaction facets of the component. It follows the Swing architecture where the developer takes care of populating the data model and the component takes care of displaying the data and enabling interactions like selection and editing.

The following figure shows the model-view architecture of JViews Gantt.



In JViews Gantt, the data model is an interface that manages activities, resources, constraints, and reservations. These objects have a set of generic properties and user-defined properties can be added to store application-dependent information. Based on this data model, JViews Gantt knows how to display the chart and how to manage end-user interactions.

The data model needs to be populated with application data. To do this, you have the choice between using a prebuilt implementation such as the in-memory data model or connecting the chart directly to your data by implementing the data model interface. The latter solution avoids data duplication and enables finer synchronization between the chart and the data.

JViews Gantt provides data sources to populate your diagram from XML files, JDBC connections, or flat files in formats like CSV (comma-separated values).

The objects that make it possible to integrate your data into the time and activity bar charts provided by JViews Gantt are activities and resources. Dependencies between activities are shown through constraints. Resources are assigned to activities through reservations.

Activities

Defines activities in the data model.

An activity is a task to be completed. Activities are hierarchical in nature. This means that a main activity, called parent activity, can be broken down into subactivities. A subactivity, or child activity, can in turn be a parent activity of other subactivities or, if it is at the very end of a branch of the hierarchy, it is a leaf activity. A leaf activity is an activity that has no child activity.

In addition to its name and identifier, an activity is defined by its start time and end time, which determine an interval called the duration of the activity. If the start and end times are identical, the duration is equal to `0`. A zero-duration activity is commonly called a milestone. Typically, milestones are not rendered by the same renderer as activities with a non-zero duration.

Resources

A resource is a means by which an activity can be completed. Resources can be persons, premises, equipment, and so forth. Like activities, resources are also hierarchical in nature.

If resources are people, the parent resource is a department while the child resources are the individual employees. Likewise, you may want to group resources by physical location or by type of machinery.

Constraints

A constraint is a type of condition set between two activities. Constraints are represented by arrowed polyline links.

Constraints can have one of the following types:

- start to start
- start to end
- end to start
- end to end

The source activity—that is, the activity whose start or end controls the start or end of another activity—is called the From activity. Conversely, the target activity—that is, the activity whose start or end depends on the start or end of another activity—is called the To activity.

Reservations

When a resource is assigned to an activity, this assignment is called a reservation. In the terminology of JViews Gantt, a reservation represents the assignment of *one* resource to *one* activity.

An activity can have multiple resources reserved and similarly, a resource can be reserved for more than one activity. The activity that reserves the resource cannot be changed after the reservation is created.

Styling

Besides displaying activities and resources, your application will need to convey qualitative information on your business data.

JViews Gantt makes use of a powerful model-based styling mechanism that relies on style rules. Each style rule defines conditions on the data model that trigger graphical changes in the display. For example, you can define a rule that draws an activity bar shape with foreground color beige and label color black when the `state` is `activity:leaf`; and another rule that turns the foreground color to gray and the label color to red when the `state` is `activity:leaf` and the `tags` property for user-defined type is `critical`.

When you define a notation, you create the basic symbols to be used for default situations. Each basic symbol is defined by at least one style rule.

Once the basic symbols are created, you then need to modify or annotate them to reflect specific properties that you want to display. For each situation, you define a rule that may complement or override the graphics effects defined by more generic rules.

The styling mechanism is also used to declare and customize options to control the appearance of the chart as a whole or of its subcomponents, the table, the Gantt sheet, the time scale, the vertical grid, or the horizontal grid.

The set of style rules is stored in a style sheet and you can dynamically load a new style sheet while keeping the same data and underlying data model. This facility is useful when you need to adapt the display to a particular situation or user profile.

The style sheet syntax conforms to the CSS syntax — a Web standard — but you need not bother with the details of this syntax at this point.

JViews Gantt eases the process of defining a notation with a development tool called the Designer. Using the Designer, you define the conditions for a rule using naturalistic language and you define the styling effects of the rule through an intuitive panel, the Styling Customizer, that lets you change graphical properties. While you are modifying rules or adding new ones, you can select a rule in a tree view and see how the notation changes within a preview window.

Architecture of JViews Gantt

Describes the architecture of JViews Gantt.

In this section

Overview

IBM® ILOG® JViews Gantt is based on the Swing variant of the Model-View-Controller model called Separable Model Architecture. In this design, the model manages the data or the values represented by the chart component, while the view manages the graphical representation of the model and handles interactions with it.

The use of this design in JViews Gantt allows you to have a clear distinction between the data model that handles the sets of data through data sources and data display that draws the graphical representation of data.

JViews Gantt features three high-level chart components, called the Gantt chart, Schedule chart, and Resource Data chart. The chart components encapsulate the Gantt library API and provide a high-level interface to its capabilities. For convenience, the chart display components are provided in the form of Java Beans®.

Binding a chart Bean to the data model enables the chart to display the contents of the data model.

The data model

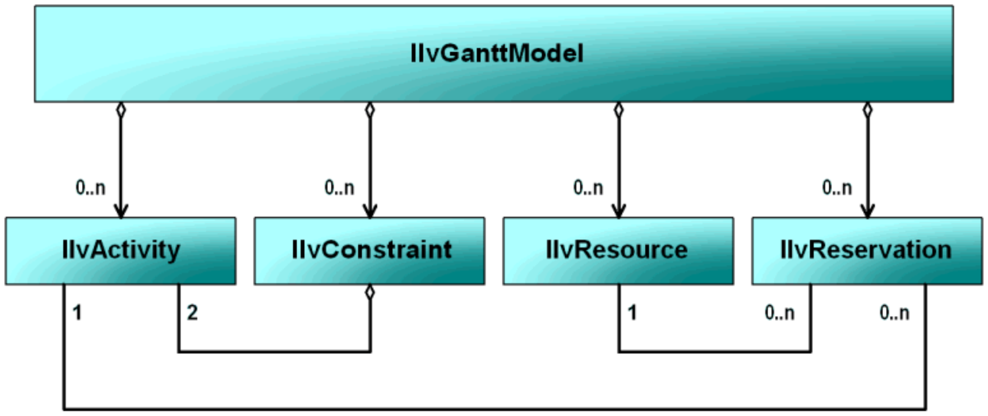
JViews Gantt provides a library of classes for displaying an abstract data model of scheduling information as a Gantt chart, a Schedule chart, or a Resource Data chart.

The scheduling data displayed by the Gantt chart, Schedule chart, and Resource Data chart is defined by the abstract `IlvGanttModel` interface. This interface defines the overall data model, which is completely abstract. This interface acts as an intelligent container for four other abstract interfaces that represent the scheduling data itself: `IlvActivity` , `IlvConstraint`, `IlvResource` , and `IlvReservation`.

The following table gives a brief description of each of the JViews Gantt data model interfaces.

Data Model Interface	Description
`IlvGanttModel`	Defines the overall JViews Gantt data model and is a container for the other four entities. See *Data* for the concept of the Gantt data model.
`IlvActivity`	Represents an activity or task that must be completed in the schedule. See *Activities*.
`IlvResource`	Represents a resource that can be allocated to an activity to enable its completion. See *Resources*.
`IlvConstraint`	Represents an activity-to-activity scheduling constraint. See *Constraints*.
`IlvReservation`	Represents the allocation of a resource to an activity. See *Reservations*.

The following figure shows the relationships between the interfaces that compose the JViews Gantt data model.



All the data model interfaces and supplied implementations are independent of the exact implementation of the other parts of the data model. For example, a default Gantt data model object can store your own custom activity implementation as easily as it would an instance of one of the supplied activity implementations. This allows you to customize only those parts of the data model that are necessary for your particular application.

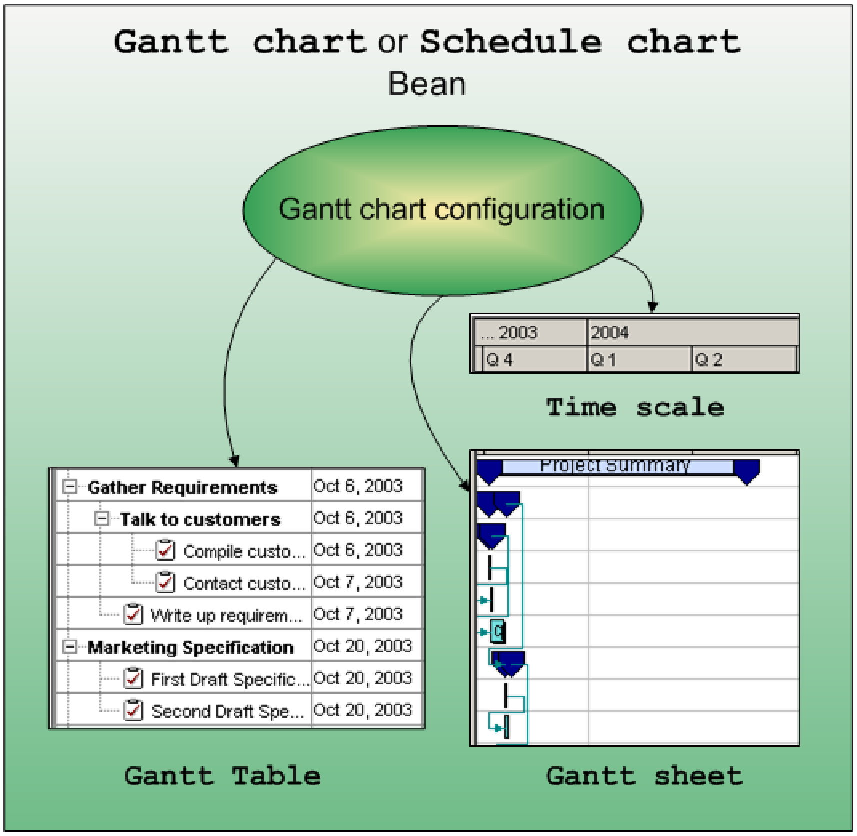
Graphical Representation

JViews Gantt features high-level Beans, called Gantt chart Bean, Schedule chart Bean, and Resource Data chart Bean. Their API is based on the classes `IlvGanttChart`, `IlvScheduleChart` , and `IlvResourceDataChart`.

The Beans encapsulate the Gantt library. Together with the Gantt data model, these Beans make up the main classes for handling Gantt, Schedule, and Resource Data charts in JViews Gantt. Chart Beans have a similar architecture and have many properties and attributes in common.

The Gantt and Schedule charts are implemented by the `IlvGanttChart` and `IlvScheduleChart` classes, respectively, both subclasses of `IlvHierarchyChart`. The Gantt chart and Schedule chart Beans can be described as a predefined combination of a configuration, a table, a Gantt sheet, and a timescale. The Gantt configuration coordinates the display of the predefined user-interface components. With the JViews Gantt SDK you can rearrange, extend, or add to your Gantt display based on the predefined Bean.

The following figure shows how the Gantt chart and Schedule chart Beans can be described as a predefined combination of a configuration, a table, a Gantt sheet, and a timescale.

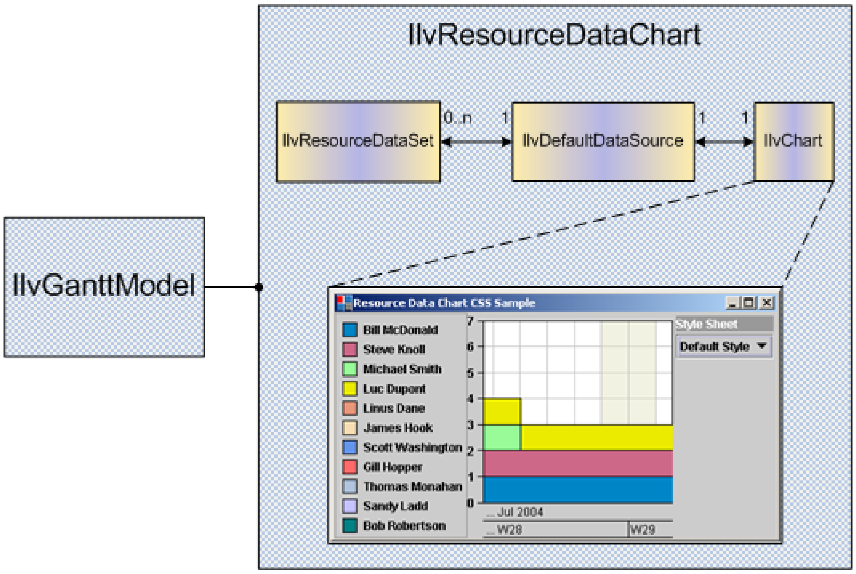

Gantt chart or Schedule chart Bean
Gantt chart configuration
... 2003
2004
Q 4
Q 1
Q 2
Time scale
Project Summary
Gather Requirements | Oct 6, 2003
Talk to customers | Oct 6, 2003
Compile custo... | Oct 6, 2003
Contact custo... | Oct 7, 2003
Write up requirem... | Oct 7, 2003
Marketing Specification | Oct 20, 2003
First Draft Specific... | Oct 20, 2003
Second Draft Spe... | Oct 20, 2003
Gantt Table
Gantt sheet

The Resource Data chart Bean is based on the `IlvResourceDataChart` class, which is a subclass of `IlvScheduleDataChart`. The Bean encapsulates the Gantt and Charts libraries. Together with the Gantt data model, the Bean is the main class for handling Resource Data charts in JViews Gantt.

The Resource Data chart displays numerical information derived from the resources in a Gantt Data model. The default data displayed by the chart is the number of activities reserved by a resource at each point in time. We call this data resource loading. The Resource Data chart displays the numerical information as a standard Cartesian chart, where the x-axis represents time. The Resource Data chart provides an alternative view of the data contained in a Gantt data model and complements the displays provided by the Gantt chart and Schedule chart beans.

The Resource Data chart makes use of the rendering capabilities of the `IlvChart` class from the IBM® ILOG®JViews Charts library. The `IlvChart` class is encapsulated by `IlvResourceDataChart`, which exposes a relevant subset of the API of `IlvChart`. See the documentation of IBM® ILOG® JViews Charts.

The following figure shows the basic architecture of the `IlvResourceDataChart` Bean.



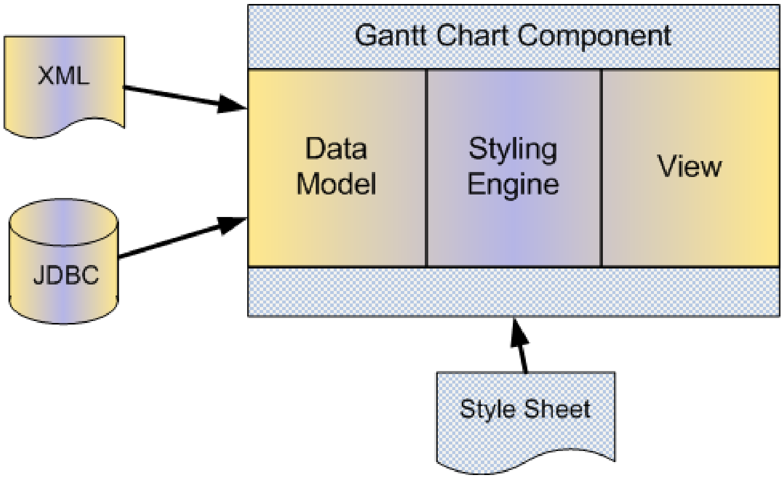
The Style Sheet

The appearance of a chart can be dynamically controlled with cascading style sheets (CSS).

CSS are defined by a W3C Recommendation and provide a powerful mechanism to customize HTML rendering inside a Web browser. In JViews Gantt the CSS level 2 Recommendation (CSS2) is adapted to the Java® language and is used to set Bean properties according to the Java object hierarchy and state. The CSS selector was designed to match HTML or XML documents, but it can be used to match a hierarchy of Java objects accessible from a model interface. The declarations are then sorted for the model objects and used according to the application that controls the CSS (or Styling) engine.

In JViews Gantt, the CSS or Styling engine is responsible for creating and customizing chart components and the graphical representation of the data at load time. At run time, the engine customizes the graphical representation of the data according to changes in the model. See Styling in *Developing with the SDK*.

The following figure shows the architecture of JViews Gantt.



When you create a chart, you can define your own style rule for:

- Chart Component

  The CSS rules are used to customize the overall appearance of the chart. See Customizing Gantt chart Options in *Using the Designer*.

- Chart Data

  The CSS rules are used to control how individual activities or constraints are rendered. See Customizing Gantt chart Options in *Using the Designer*.

Developing with JViews Gantt

Shows the overall process flow for building a chart component.

In this section

Overview

You can develop a chart component completely in the Designer or with the SDK. Typically, you would develop your chart in the Designer and then add graphic refinement in the SDK if you need more sophisticated features that are not currently available in the Designer. Resource Data charts can only be developed by using the SDK.

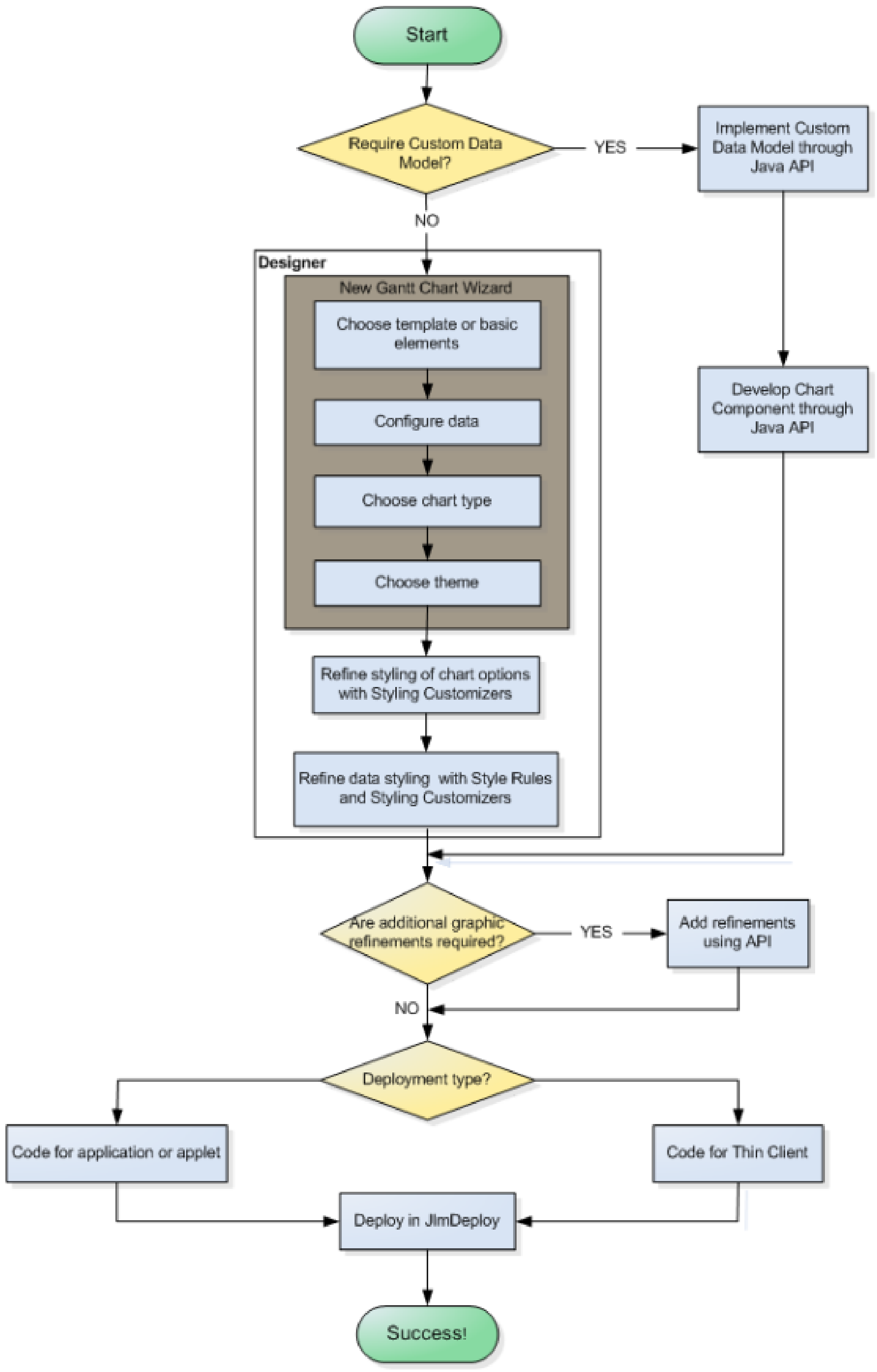
The process flow

The main phases in the recommended process of developing a chart with JViews Gantt are:

1. Populating the data model with your application data.
2. Creating a chart with the Designer.
3. Developing the Java® Swing application, applet, or servlet that uses the chart component.
4. Customizing the application.

When you start a development project, you usually go sequentially from one phase to the other to create a first prototype. Once the prototype is ready, you will probably concentrate on each phase in turn, using a classical incremental development approach.

The following figure shows the process flow for building a chart component in IBM® ILOG®JViews Gantt.


Start
Require Custom Data Model?
YES
Implement Custom Data Model through Java API
NO
Designer
New Gantt Chart Wizard
Choose template or basic elements
Configure data
Choose chart type
Choose theme
Develop Chart Component through Java API
Refine styling of chart options with Styling Customizers
Refine data styling with Style Rules and Styling Customizers
Are additional graphic refinements required?
YES
Add refinements using API
NO
Deployment type?
Code for application or applet
Code for Thin Client
Deploy in JlmDeploy
Success!

The figure presents a high-level view of the overall process.

The detailed tasks belonging to each step are described in the user documents:

- Using the Designer
- Developing with the JViews Gantt SDK

Unless otherwise specified, all the steps of the process flow can be carried out through the Designer and are documented in Using the Designer.

You can develop a Gantt chart from scratch or by connecting to existing data, with the Designer. Working with the Designer, you develop components entirely through the GUI of the Designer. The only code you need to write is to integrate your component into an application or to deploy an application as a thin client.

Populating the data model

The data model is the source of the display: JViews Gantt shows activities, constraints, resources, and reservations to represent the data model with a look-and-feel defined in the style sheet. Without a data model, you cannot define a style sheet and you cannot create a chart.

Prototyping

If you are starting a project and you want to prototype your application rapidly, you will probably adopt the fastest way of populating the data model.

JViews Gantt offers a prebuilt in-memory data model that you can populate manually in the Designer or programmatically, or that can be populated from XML or text (CSV) files through *data sources*. Another prebuilt data model is available which can be populated from a database using the JDBC data source. If you can use JDBC, XML, or a flat file, you can directly import your data into the Designer.

There are two cases:

- ♦ If your application data is compliant with one of the JViews Gantt data sources, you can easily populate the data model using the data source facility in the Designer or by using the API.
- ♦ If your application data is not compliant with one of the JViews Gantt data sources, you should probably create a sample of your data in an XML file or in Microsoft® Excel, to emulate your data.

As soon as you can populate the data model, you can display the chart using the default style sheet. Then you can define your own styling to create business-specific representations.

If you use a flat file or data from a database that can be connected through JDBC, you must be able to map your data to the objects required by the default data model of IBM® ILOG® JViews Gantt. See *Data from a database*.

If your data is in XML, it must be in the format of the Schedule Data eXchange Language (SDXL) or be made to conform to this format. See *XML data*.

If you have an in-memory data model, you need to be willing to replicate the data by copying it to the default data model. See *In-memory data*.

Otherwise, you must undertake some customization using the Java® API of the SDK for extending or developing the component. See Connecting to data in-memory in *Developing with the SDK*.

To be able to use the default data model, you need, at the very least, to be able to map to the object that represents activities.

Development

When you develop the operational application, you can still use the in-memory data model and one of the data sources provided with JViews Gantt. This may imply that your application can generate an XML file that JViews Gantt can read, or that you use XSLT to transform the XML format of your application.

If your application data is in an object model, you can implement the data model interface of JViews Gantt to map your object model closely to the JViews Gantt data model. This approach requires more development, but it prevents any data duplication, and ensures perfect synchronization between the data and the graphics.

Data from a database

If you can map your database data directly to the objects required by the data model in IBM® ILOG® JViews Gantt, you can write SQL queries and map to columns in the New Gantt Chart Wizard of the Designer. You must be able to map to the object representing activities at the very least. The other objects in the data model represent resources, constraints, and reservations. The mapping is to objects in the normal JDBC data model used to connect to a database in IBM ILOG JViews Gantt

.

Note: IBM ILOG JViews Gantt provides predefined SQL queries for Microsoft® Project databases.

If you cannot map your database data directly to the IBM ILOG JViews Gantt objects, you can still use the normal JDBC data model in the product, but you will have to map the objects yourself using Java® code in the SDK. See *Developing with the SDK*. Examples of database data that cannot be mapped directly are when activities are spread over several tables or when the data requires advanced mathematical calculations that cannot be handled in SQL.

XML data

If your XML data is already in the format of the Schedule Data eXchange Language (SDXL), you can use the normal IBM® ILOG® JViews Gantt reader and the default data model. You can do this through the New Gantt Chart Wizard in the Designer. The SDXL format is compatible with the format used by IBM ILOG Gantt for .NET.

If your XML data is in a different format, you can use XSLT to put your data into SDXL format before you import the data. You can change the format using the New Gantt Chart Wizard in the Designer. Then you can use the normal reader and default model.

If the transformation of your XML data requires features that XSLT cannot handle, such as advanced math., you must implement your own reader in Java® code by extending the component in the SDK and plug the reader into the default model. See How to read an IlvGanttModel from an SDXL file using serialization in Developing with the SDK.

In-memory data

If you do not want to replicate the data from the in-memory model to the default model, you have the following choices:

♦ If your model is based on four Swing table models (`TableModel`), with one for activities, one for resources, one for constraints, and one for reservations, you can probably use the normal Gantt table model directly with a suitable mapping configuration.

- If your model is of a different type, you must write your own version of the data model used by IBM® ILOG® JViews Gantt to wrap your existing model.

If you are willing to copy your data, you can populate the default data model used in IBM ILOG JViews Gantt with data from your in-memory data model.

All these solutions require using some Java® code in the SDK. See Connecting to data in-memory in *Developing with the SDK*.

Creating a Gantt chart with the Designer

The default style sheets provided with JViews Gantt display a classical activity-oriented or resource-oriented Gantt chart.

You can customize the representation with the JViews Gantt styling facilities.

The easiest way to define the chart look-and-feel is to use the Designer. You can also write your style sheet manually without the Designer and set up the chart parameters with the SDK, but you should try the Designer first before deciding to take a different route.

How the Designer helps the developer

As a Java® developer, you can decide to build chart components with the Java API of IBM® ILOG® JViews Gantt and the CSS2 styling language. Starting with a point-and-click editor like the Designer will make the early phases of development easier and faster .

The Designer is a development tool that is designed for defining the look-and-feel of your diagram.

With the Designer:

- ♦ You do not need to know the CSS syntax (although you do need to know the principles behind CSS)
- ♦ You can change the graphics properties through simple dialogs
- ♦ You get instant feedback on the modifications you make on style rules
- ♦ You can modify the data model
- ♦ You generate a project that can be used right away in your application

The Designer offers facilities for filling the data model, viewing the Gantt chart, viewing and creating style rules, and saving a project.

See the gallery of Designer projects supplied in the JViews Gantt samples at:

`<installdir>/jviews-gantt86/samples/gallery`

Filling the data model

To use the Designer, all you need is a data model. The section *Populating the data model* explains considerations on how to populate a data model.

The Designer provides a wizard that helps you to fill the data model from XML files, a JDBC connection, CSV files, or by entering the data manually. It also provides samples of models to help you get started.

Viewing the data model as a chart

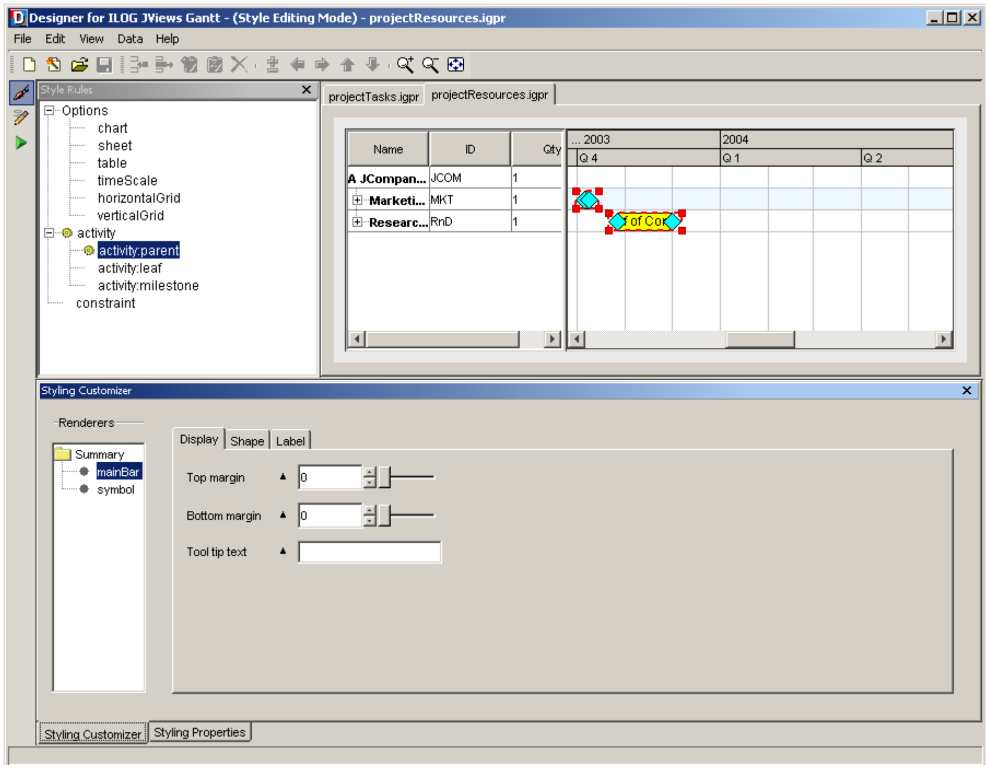
Once the data model is filled, you can keep the default style sheet or select another style sheet to display the chart. At this stage, you can preview your chart in the Designer.

Viewing Style Rules

Once you have chosen a style sheet, you can start to create style rules to refine the graphical representation of your data model.

The following figure shows how the Designer presents the style rules in a tree, from which you can manage and edit rules.

The following figure shows the Style Rule hierarchy in left pane of Designer main window.



Top margin 0

Bottom margin 0

Tool tip text

Each rule displays its conditions in the tree so that you clearly see which objects in the data model are matched by the rule. If the conditions become too long for easy reading, you can define custom rule names instead.

The graphical properties set by the rule are displayed in a panel called the Styling Customizer.

Creating Style Rules

When you create a new rule, a naturalistic language editor helps you to specify the selector part by proposing conditions based on your data model.

Once the conditions are defined, you use the Styling Customizer panel to specify the graphic effects to display when the conditions are satisfied for an activity or a constraint. You can just change graphical properties like color or shape, or you can add new shapes, texts, or images as elements of a composite graphic.

Saving the project

When you have set up the look-and-feel of your chart, you can save your project to be loaded into your application.

See the gallery of Designer projects supplied in the JViews Gantt samples at:

`<installdir>/jviews-gantt86/samples/gallery`

Developing the application

At some point you need to develop one of the following types of application: a Java® Swing application, an applet, or a servlet. In this application you create an `IlvGanttChart` or `IlvScheduleChart` object; then, you load the project generated by the Designer.

Similarly, in an application that uses a Resource Data chart, you create an `IlvResourceDataChart` object, but you must bind the chart to the data model by code and code the styling of the chart and its data. See Basic steps in using the Resource Data chart bean - details in *Developing with the SDK*.

For prototyping, you can decide to start with the prebuilt JViews Gantt application that only requires a loaded project to run.

See the gallery of Designer projects supplied in the JViews Gantt samples at:

`<installdir>/jviews-gantt86/samples/gallery`.

Customizing the Application

You may well have precise requirements for your application that cannot be satisfied just with the style sheets and the prebuilt behavior of JViews Gantt. This is the case if your application requires specialized interactions or if you had to implement the data model interface to connect to your application data.

With the JViews Gantt SDK, you have a comprehensive API to use or extend the Java® classes involved in the creation of your chart. You can basically access all the entities that play a role in and around the chart: the data sources, the renderers, the views, the interactors, the composite graphics, and the chart objects themselves.

Decision Points

The only decisions you need to make about whether to use the Designer or the SDK concern the type of data you use, the type of chart you want to create, or the extent of graphic refinement that you want to incorporate.

If your data cannot be connected by JDBC, is not in XML or a flat file (that uses the separators comma, tab, semicolon, or space), or is not held in an in-memory data model, you must use the Java® API. You can, however, still write your CSS style sheet in the Designer.

You cannot develop a Resource Data chart in the Designer. You must use the SDK.

If after you complete a development cycle in the Designer you find that you require graphic refinement features that are not available in the GUI, you can achieve these by using the Java API directly.

If you decide to write your own renderer with the Java API, you may want to use even more advanced features available through the Java API of the JViews Framework.

In any case, you can carry out most of your development in the Designer.

Index