Introduction to FlexGantt

Topic: Components

Dirk Lemmermann
Software & Consulting
Zurich, Switzerland

Content

- Gantt Chart / Dual Gantt Chart
- Layer Container
- Tree Table (sorting, filtering)
- Row Headers (Tree Table / Layer Container)
- Timeline, Dateline, Eventline
- Utility & Navigation Control Panel
- Selectors

Gantt Charts

AbstractGanttChart

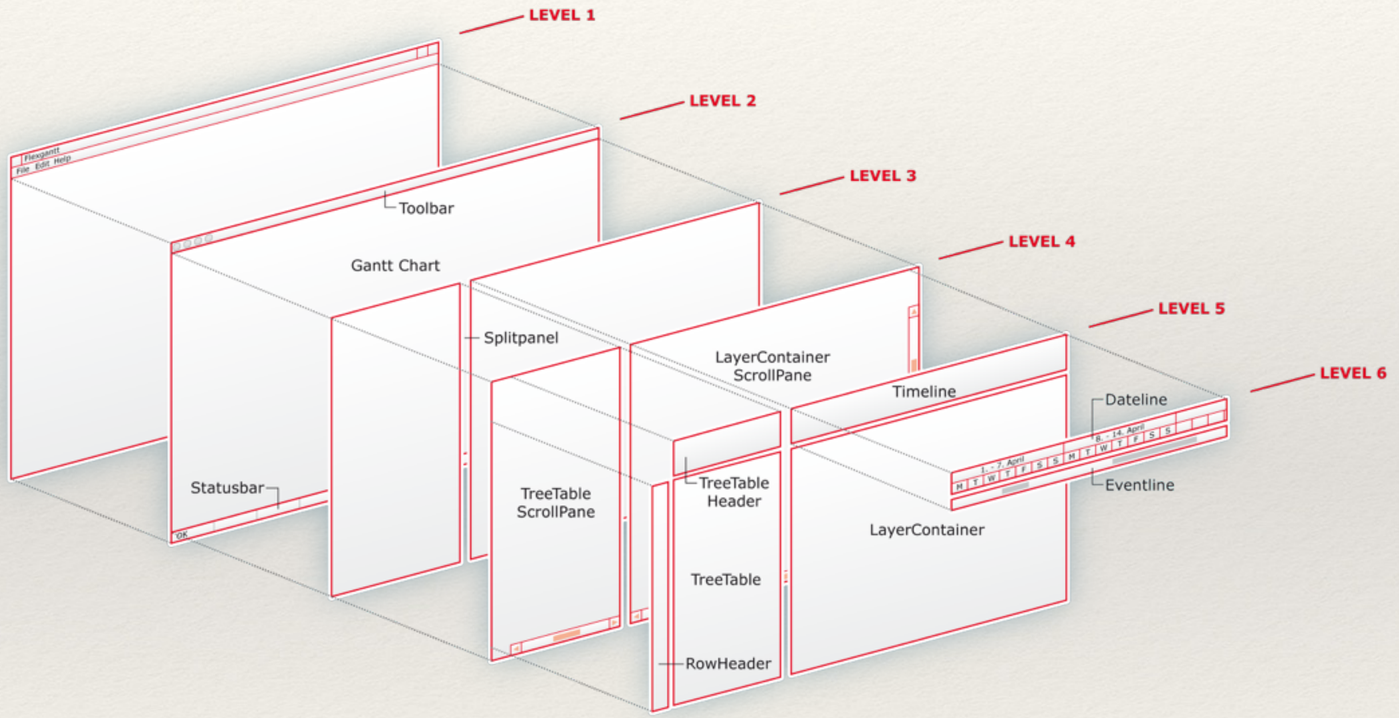
- Carries almost all functionality.
  - Creates sub-components via pluggable component factory
  - Controls visibility of all layers (e.g. setPopupVisible()).
  - Utility methods to control the horizon / overall time span.
  - Column model (for tree table).
  - Paging model.
  - Command stack (commandExecute/Undo/Redo()).
  - Command interceptors.
  - Messages (addMessage(IMessage)).

Gantt Chart

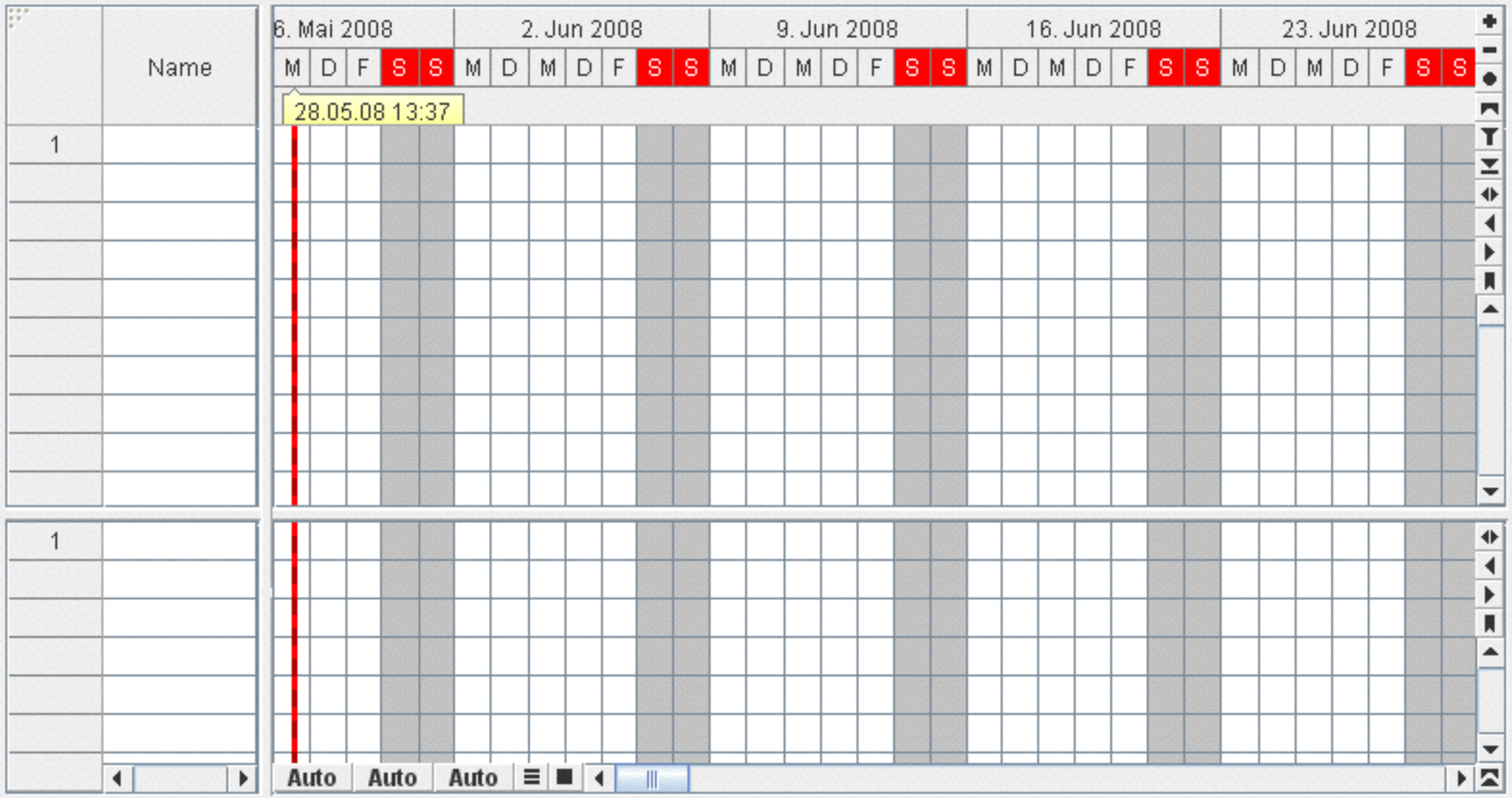

Name
6. Mai 2008
2. Jun 2008
9. Jun 2008
16. Jun 2008
23. Jun 2008
M D F S S M D M D F S S M D M D F S S M D M D F S S M D M D F S S
28.05.08 13:40
1
Auto
Auto

COMPONENTS

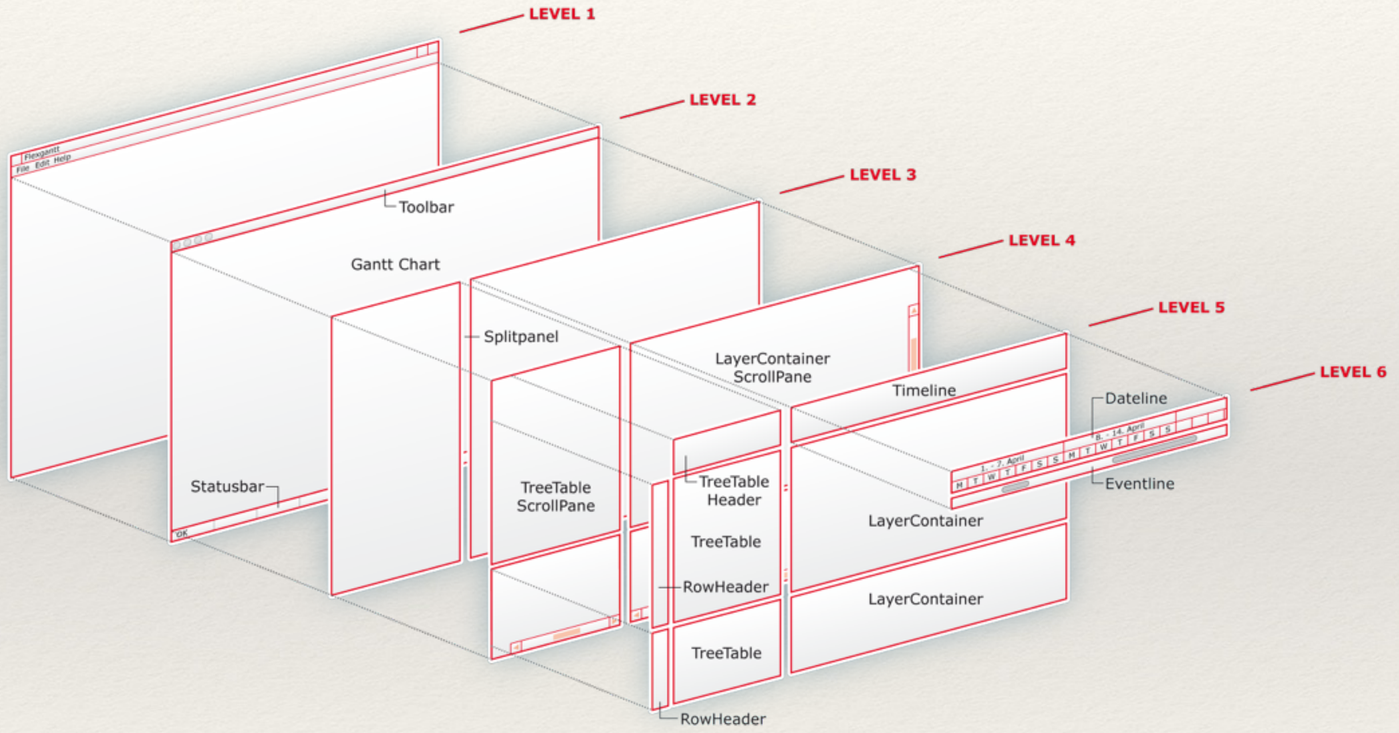
Gantt Chart



DualGanttChart



Dual Gantt Chart



Subclassing / Customizing Components

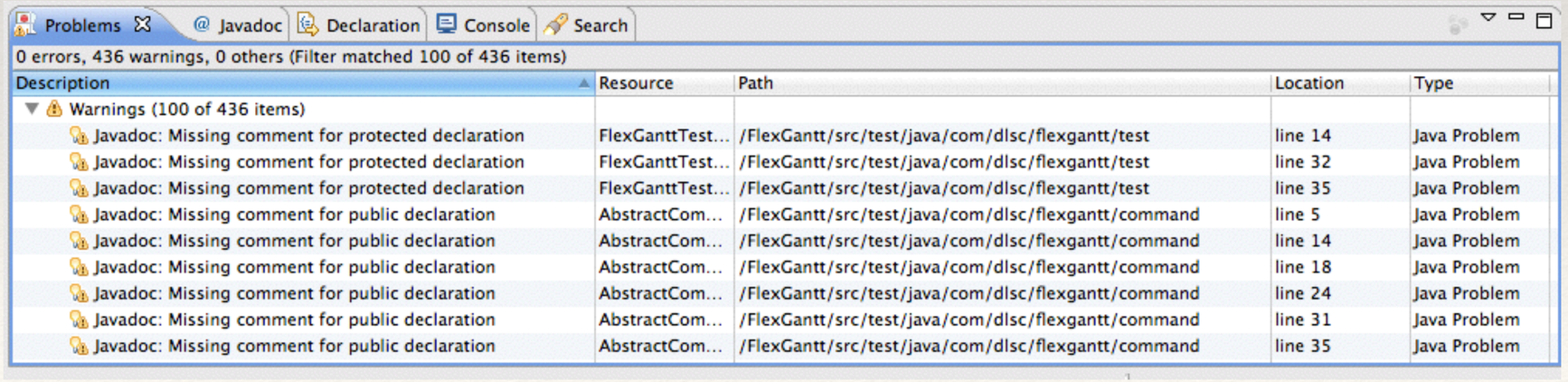
- Implement your own component factory and pass it to the Gantt chart constructor.
- Subclass DefaultComponentFactory.
- Override appropriate component factory method.
- Call super.createXYZ() if you just want to perform some settings or ….
- …. create your own subclass of the XYZ component and return it.

ComponentFactory

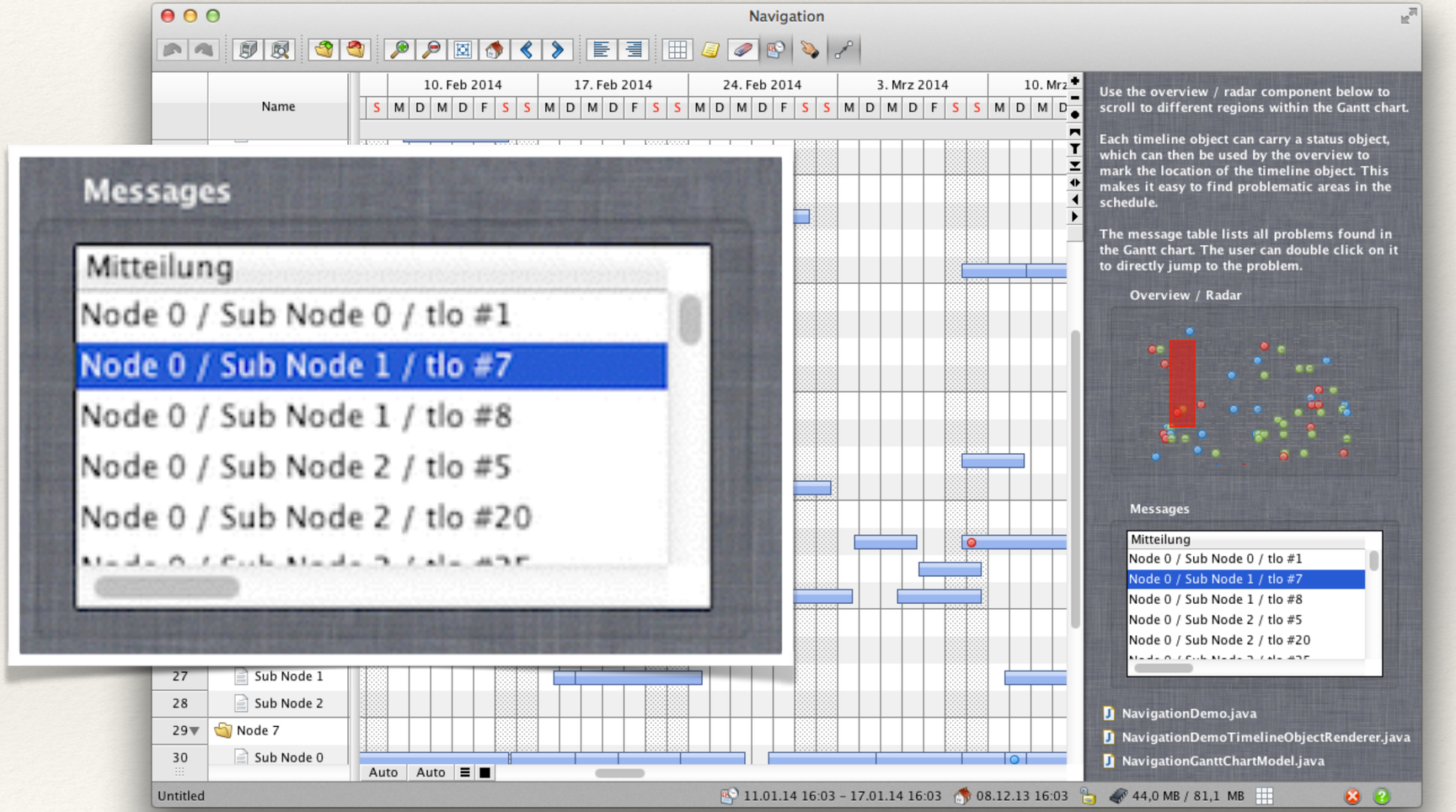
```
public interface IComponentFactory {

    Timeline createTimeline(AbstractGanttChart gc);

    Dateline createDateline(Timeline timeline);

    Eventline createEventline(Timeline timeline, Dateline dateline);

    TreeTable createTreeTable(AbstractGanttChart gc, ITreeTableModel model);

    TreeTableRowHeader createTreeTableRowHeader(TreeTable table);

    LayerContainer createLayerContainer(AbstractGanttChart gc, TreeTable table, IGanttChartModel model);

    LayerContainerRowHeader createLayerContainerRowHeader(LayerContainer lc);

    …
}
```

Messages

- Messages can be attached to the Gantt chart.
- Similar to warnings / errors shown in Eclipse view.
- AbstractGanttChart.addMessage(IMessage): TreePathMessage, TimelineObjectPathMessage, RelationshipMessage
- Double-click takes the user to the problematic area.



Messages


Navigation
Name
10. Feb 2014
17. Feb 2014
24. Feb 2014
3. Mrz 2014
Messages
Mitteilung
Node 0 / Sub Node 0 / tlo #1
Node 0 / Sub Node 1 / tlo #7
Node 0 / Sub Node 1 / tlo #8
Node 0 / Sub Node 2 / tlo #5
Node 0 / Sub Node 2 / tlo #20
Use the overview / radar component below to scroll to different regions within the Gantt chart.
Each timeline object can carry a status object, which can then be used by the overview to mark the location of the timeline object. This makes it easy to find problematic areas in the schedule.
The message table lists all problems found in the Gantt chart. The user can double click on it to directly jump to the problem.
Overview / Radar
Messages
Mitteilung
Node 0 / Sub Node 0 / tlo #1
Node 0 / Sub Node 1 / tlo #7
Node 0 / Sub Node 1 / tlo #8
Node 0 / Sub Node 2 / tlo #5
Node 0 / Sub Node 2 / tlo #20
NavigationDemo.java
NavigationDemoTimelineObjectRenderer.java
NavigationGanttChartModel.java
27 Sub Node 1
28 Sub Node 2
29 Node 7
30 Sub Node 0
Auto
Auto
Untitled
11.01.14 16:03 – 17.01.14 16:03
08.12.13 16:03
44,0 MB / 81,1 MB

Layer Container

LayerContainer

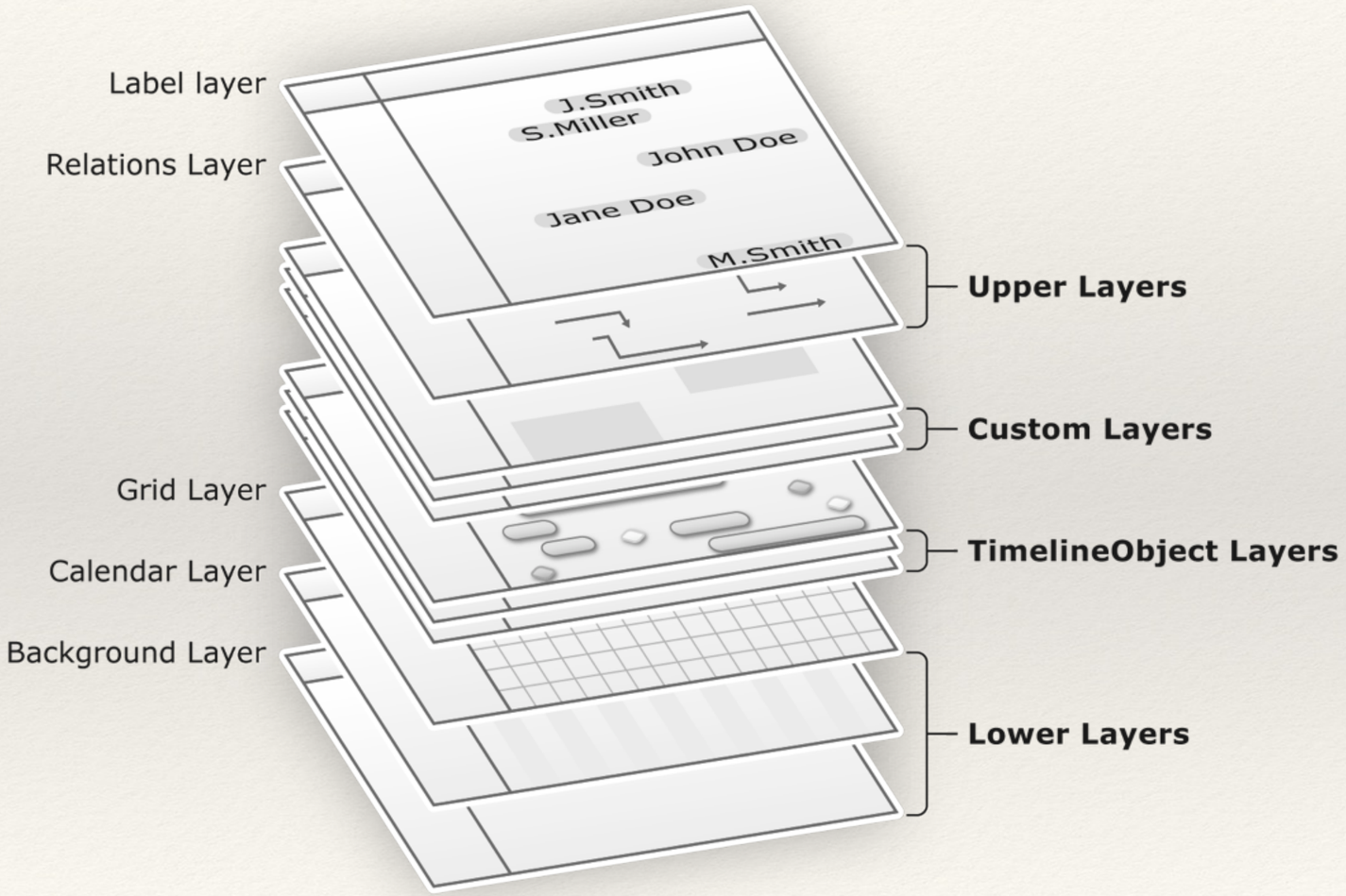
- Responsible for displaying the various layer types:
  - System Layer, Timeline Object Layers, Custom Layers.
  - Layer functions: show / hide / to front / to back.
  - Stores the timeline object layer selection models.
  - Navigation functions: show all, earliest, latest TLOs.

LayerContainer

- Determines the order of the system layers.

```
protected List<Class<? extends AbstractSystemLayer>>
    getTypesOfLowerSystemLayers();

protected List<Class<? extends AbstractSystemLayer>>
    getTypesOfUpperSystemLayers();
```

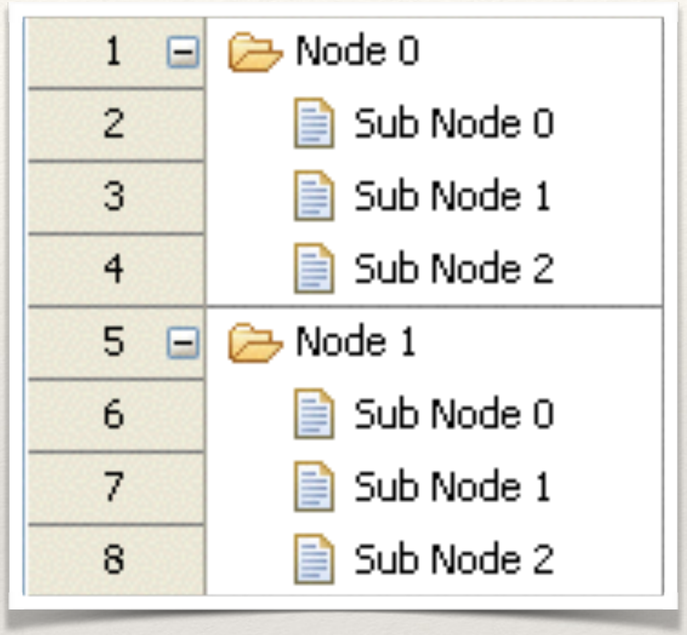
Layer Stack


Label layer
Relations Layer
Grid Layer
Calendar Layer
Background Layer
J.Smith
S.Miller
John Doe
Jane Doe
M.Smith
Upper Layers
Custom Layers
TimelineObject Layers
Lower Layers

Tree Table

Tree Table

- The left-hand side of each Gantt chart is a tree table.
- The tree table is the „master“ controlling the layer container.
- Not Swing table or Swing tree.
- Uses Swing concepts: pluggable renderers and editors.
- Invokes policies to control its behavior.
- Uses a central column model.



Tree Table: Filtering

```
/**
 * This interface can be used to filter the rows / nodes shown in the
 * tree table component.
 *
 * @see TreeTable#setNodeFilter(INodeFilter)
 */
public interface INodeFilter {

  /**
   * Determines whether the given node will be included in the tree table
   * view or not.
   */
  boolean includeNode(Object node);
}
```

Tree Table: Sorting 1

DefaultMutableTreeTableNode:

```
  public void sort(int[] modelIndices, boolean[] sortDirections) {
    this.sortModelIndices = modelIndices;
    this.sortDirections = sortDirections;
    if (children != null) {
      Collections.sort(children, this);
      for (Object obj : children) {
        ITreeTableNode child = (ITreeTableNode) obj;
        child.sort(modelIndices, sortDirections);
      }
    }
  }
```

Tree Table: Sorting 2

DefaultMutableTreeTableNode (implements Comparable):

```
    public int compare(Object o1, Object o2) {
        ITreeTableNode node1 = (ITreeTableNode) o1;
        ITreeTableNode node2 = (ITreeTableNode) o2;
        for (int i = 0; i < sortModelIndices.length; i++) {
            int modelIndex = sortModelIndices[i];
            boolean ascending = sortDirections[i];
            Object value1 = null;
            Object value2 = null;
            if (modelIndex == KeyColumn.MODEL_INDEX) {
                value1 = node1.getKey();
                value2 = node2.getKey();
            } else {
                value1 = node1.getColumnValue(modelIndex);
                value2 = node2.getColumnValue(modelIndex);
            }
            int result = compare(modelIndex, value1, value2, ascending);
            if (result != 0) {
                return result;
            }
        }
        return 0;
    }
```

Tree Table: Sorting 3

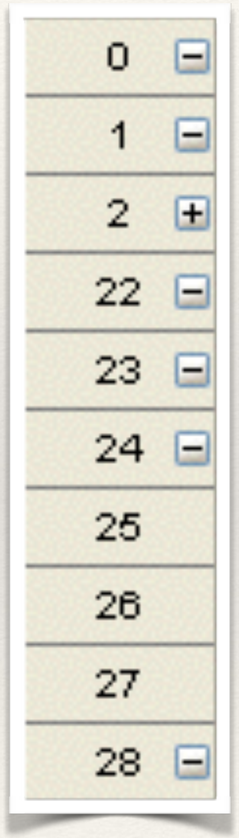
DefaultMutableTreeTableNode (implements Comparable):

```
    protected int compare(int modelIndex, Object value1, Object value2,
            boolean ascending) {

        if (value1 == null) return -1;
        if (value2 == null) return +1;

        Comparator comparator = null;
        if (modelIndex == KeyColumn.MODEL_INDEX) {
            comparator = getKeyComparator();
        } else {
            comparator = getComparator(modelIndex);
        }
        int result = 0;
        if (comparator != null) {
            result = comparator.compare(value1, value2);
        } else if (value1 instanceof Comparable) {
            result = ((Comparable) value1).compareTo(value2);
        } else {
            throw new IllegalArgumentException("unable to sort model index“);
        }
        if (!ascending) {
            if (result != 0) {
                result = -result;
            }
        }
        return result;
    }
```

Row Headers

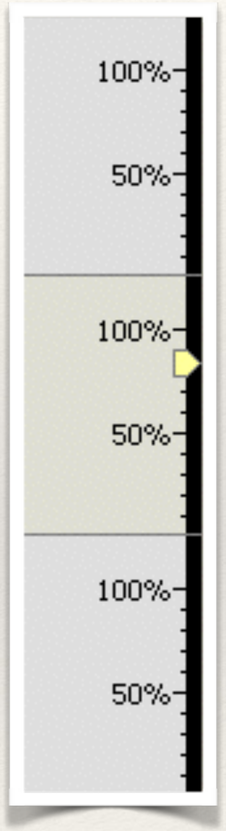
Tree Table Row Header

- A component used for displaying row numbers and expand / collapse icons on the left-hand side of the tree table.
- Can be customized to display any kind of information.
- Uses an ITreeTableRowHeaderRenderer instance to draw the cells.



LayerContainerRowHeader

- Shown on the left edge of the layer container.
- The row header can be used to visualize row specific information that is not supposed to move while the user scrolls horizontally.
- Uses ILayerContainerRowHeaderRenderer to draw the cells.



NavigationControlPanel

Navigation Control Panel

- Created by the component factory as the UPPER_RIGHT_CORNER of the layer container.
- Various control features (zoom in / out, etc…).

```
public NavigationControl getNavigationControl(NavigationControlType type);
```

Navigation Controls


Set time span / horizon


Zoom Out


Zoom In


Go to „now“


Show all objects


Go to latest objects


Go to earliest objects


Select a granularity


Go to specific time point


Bookmarks

UtilityControlPanel

UtilityControlPanel

- Created by the component factory as the LOWER_LEFT_CORNER of the layer container.
- Adjusts its controls by itself.
  - Shows paging controls if a „paging model“ is set.
  - Shows grid controls based on number of layer containers shown by the Gantt chart.

Timeline, Dateline, Eventline

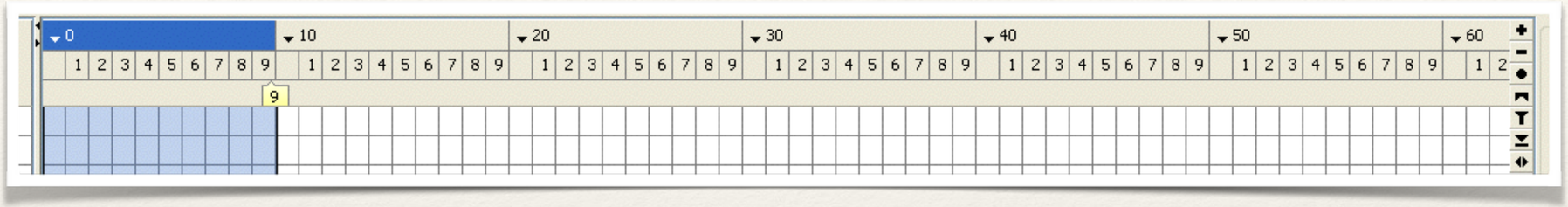
Timeline

- Parent container for Dateline and Eventline.
- Dateline displays the time (Mon, Tue, Wed, …..).
- Eventline displays „global“ activities and milestones..

Dateline

- Displays a major and a minor time granularity.
- Uses an instance of IDatelineModel to …
  - Compute grid line locations
  - Format date strings
  - Retrieve spreadsheet cell width
- Appearance customizable via pluggable dateline renderers.

SimpleDateline



- „Simply“ counts units (1, 10, 100, 1000, 10000, …)
- Used for planning, not scheduling.
- Uses an instance of SimpleGranularityDatelineModel

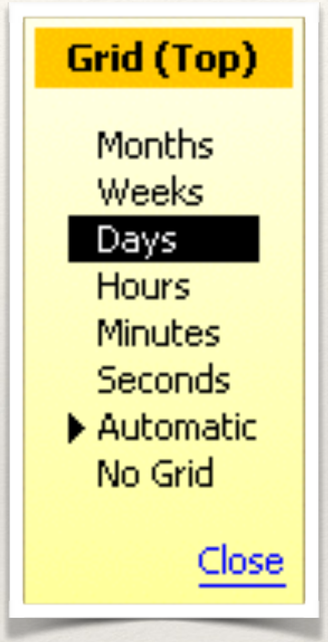
Eventline

- Displays „markers“: time at cursor location, time span during drag.
- Displays „global“ activities and milestones that are relevant for all rows in the Gantt chart (e.g. holidays).
- Eventline objects can be created and edited directly inside the eventline.
- The time span of eventline objects can be visualized by the EventlineLayer in the layer container below.

Selectors

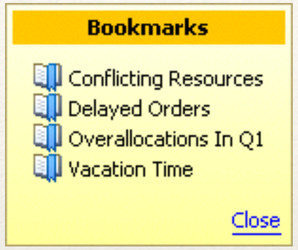
Selectors

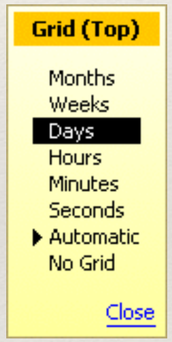
- Little slide in / slide out dialogs for quickly changing settings.
- Provided out-of-the-box, but can be customized.
- ISelectorFactory is used to produce the selectors.
- Override AbstractGanttChart.getSelectorFactory() to use custom factory.

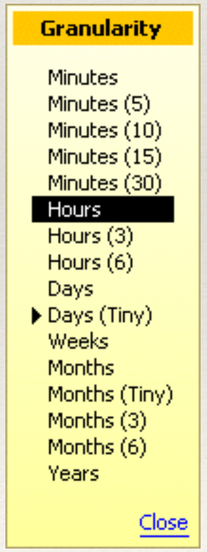


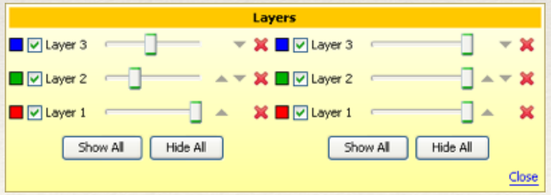
Selectors

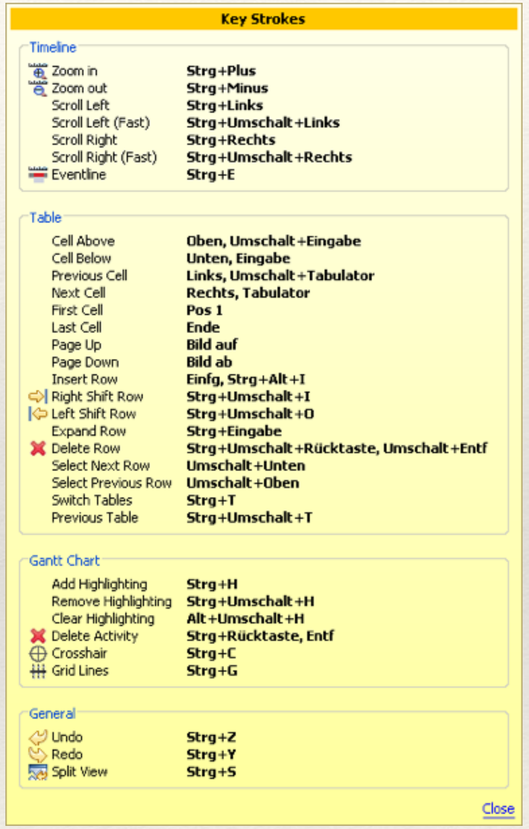
- Bookmarks
- Granularity
- Grid
- Key Strokes
- Layers











Selectors

- Overview
- Paging
- Show Time
- Columns
- Horizon