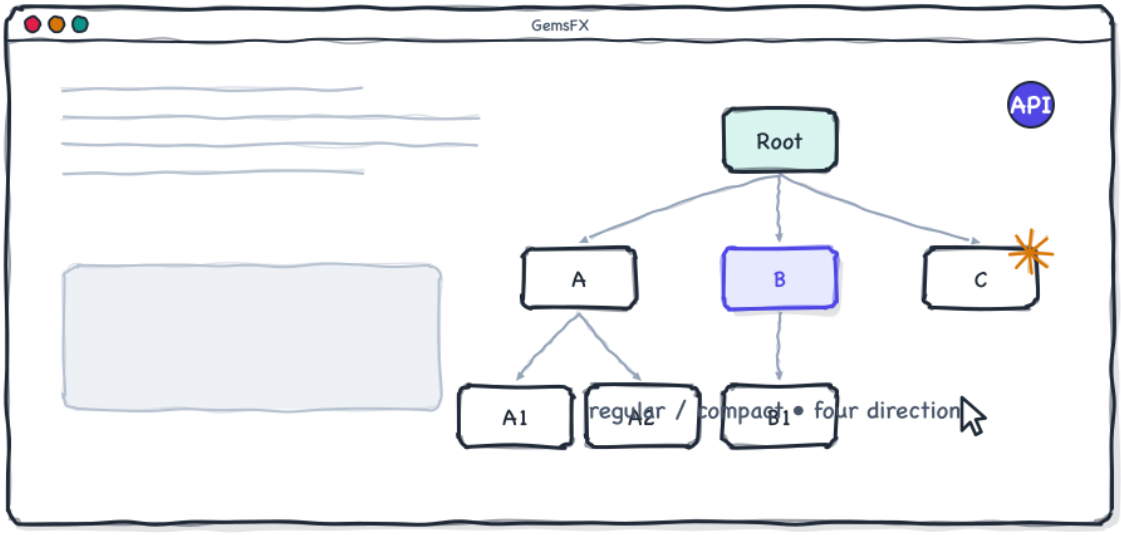
TreeNodeView

Developer Manual

GemsFX • com.dlsc.gemsfx.treeview • JavaFX Custom Controls



Generated cartoon overview of TreeNodeView.

TreeNodeView renders a TreeNode model as a node-link diagram with configurable layout direction, regular or compact layout, cell factory, spacing, alignment and link strategies.

Table of Contents

1. Introduction

TreeNodeView lives in `com.dlsc.gemsfx.treeview`. It does not use JavaFX TreeItem; instead, its model is `TreeNode`, which stores children, optional linked nodes, expansion state and optional per-node dimensions.

1.1 Key features

- Supports TOP_TO_BOTTOM, BOTTOM_TO_TOP, LEFT_TO_RIGHT and RIGHT_TO_LEFT layout directions.
- Supports REGULAR and COMPACT layout algorithms.
- Default cells are TreeNodeCell instances with label and disclosure arrow.
- Links are pluggable through LinkStrategy implementations.
- TreeNode.name creates deterministic style classes for node links.

1.2 Maven dependency

```
<dependency>
    <groupId>com.dlsc.gemsfx</groupId>
    <artifactId>gemsfx</artifactId>
    <version>4.4.1</version>
</dependency>
```

Use package `com.dlsc.gemsfx.treeview`.

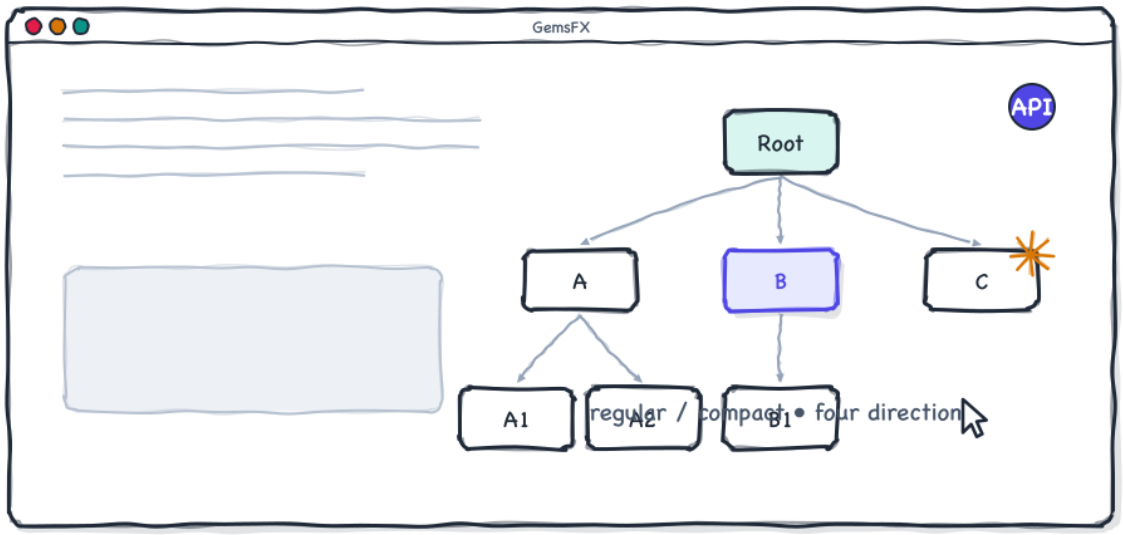
2. Getting started

The snippet below uses only APIs verified in the source and demo code.

```
TreeNode<String> root = new TreeNode<>("Root");
root.getChildren().addAll(new TreeNode<>("A"), new TreeNode<>("B"));

TreeNodeView<String> view = new TreeNodeView<>(root);
view.setLayoutDirection(TreeNodeView.LayoutDirection.TOP_TO_BOTTOM);
view.setLayoutType(TreeNodeView.LayoutType.REGULAR);
view.setLinkStrategy(new CurvedLineLink<>());
```

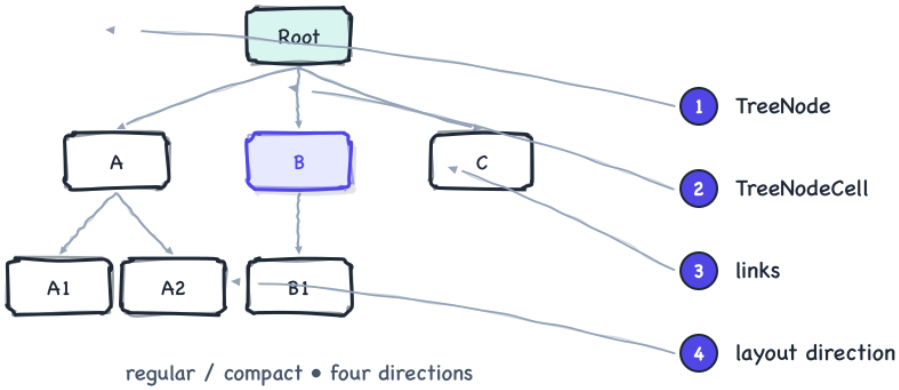
Minimal setup for `TreeNodeView`.



A first look at the control.

3. Anatomy

The diagram and table identify the nodes, model objects and style classes that matter when using or styling the control.



The main parts of the control.

Part	Type / style	Description
TreeNodeView	tree-node-view	Control root and layout properties.
TreeNode	model node	Value, children, linkedNodes, expanded, width, height and name.
TreeNodeCell	tree-node-cell	Default cell with label and disclosure arrow.
tree-content	Group	Skin content group containing cells and link nodes.

Part	Type / style	Description
LinkStrategy	link-line/path/curve/arrow	Draws parent-child and extra links.

4. Control API

4.1 Tree and cells

Property	Type	Default	Description
root	ObjectProperty<TreeNode<T>>	null	Root node; null shows placeholder.
cellFactory	ObjectProperty<Callback<T, TreeNodeCell<T>>>	TreeNodeCell::new	Creates cells for node values.
placeholder	ObjectProperty<Node>	Label "No tree root."	Shown when root is null.
refresh()	method	skin rebuild	Rebuilds visual tree from current model and properties.

4.2 Layout

Property	Type	Default	Description
cellWidth	DoubleProperty	60	Default cell width; styleable.
cellHeight	DoubleProperty	30	Default cell height; styleable.
hgap	DoubleProperty	20	Horizontal gap; styleable.
vgap	DoubleProperty	50	Vertical gap; styleable.
nodeLineGap	DoubleProperty	10	Gap between cell and connecting lines; styleable.
rowAlignment	ObjectProperty<VPos>	CENTER	Alignment for same-level nodes in vertical directions; styleable.
columnAlignment	ObjectProperty<HPos>	CENTER	Alignment for same-level nodes in horizontal directions; styleable.
layoutType	ObjectProperty<LayoutType>	REGULAR	REGULAR or COMPACT; styleable.
layoutDirection	ObjectProperty<LayoutDirection>	TOP_TO_BOTTOM	Tree growth direction; styleable.

4.3 TreeNode

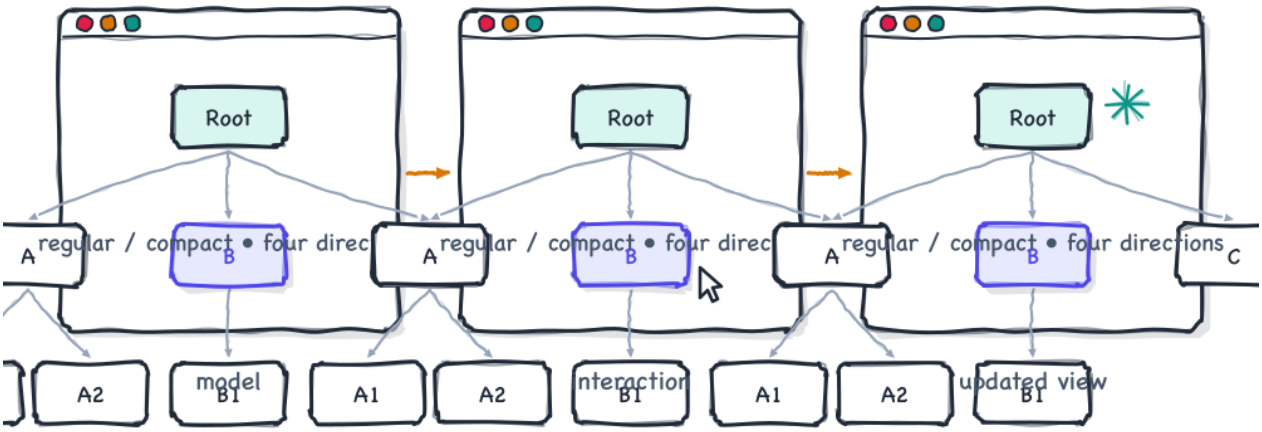
Property	Type	Default	Description
value	ObjectProperty<T>	null	User value.
children	ObservableList<TreeNode<T>>	empty	Hierarchical child nodes; parent is maintained automatically.

Property	Type	Default	Description
linkedNodes	ObservableList<TreeNode<T>>	empty	Additional non-hierarchical links.
expanded	BooleanProperty	true	Controls visibility of descendants.
width / height	DoubleProperty	USE_TREE_CELL_SIZE	Per-node dimensions; sentinel uses view defaults.
name	String	null	Optional style identifier for node/link style classes.

5. Behaviour

5.1 Layout algorithms

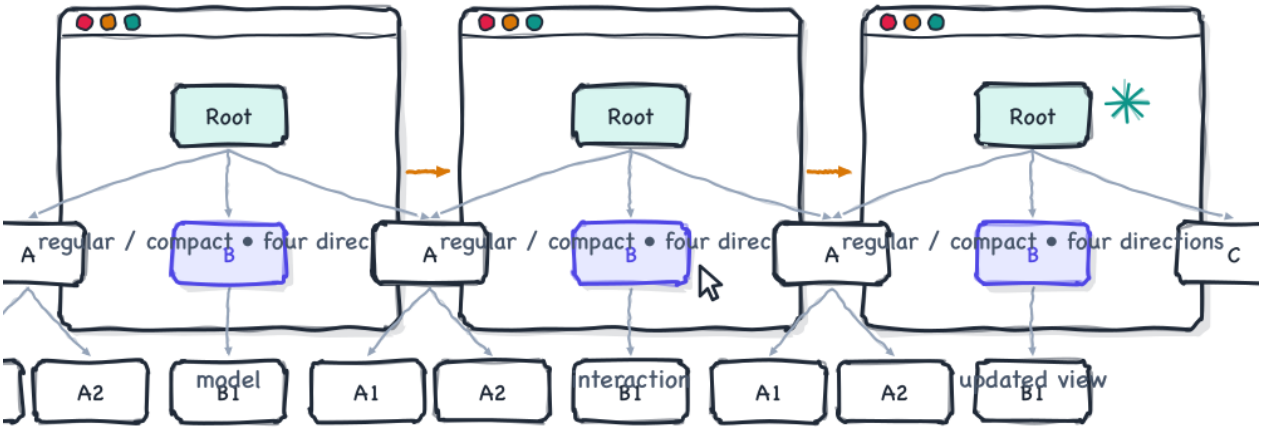
REGULAR layout sizes each subtree from child totals. COMPACT layout lays out levels breadth-first using the widest row or tallest column, which can reduce space.



The main runtime behaviour.

5.2 Expansion and rebuilding

TreeNodeCell binds its expanded property bidirectionally to TreeNode.expanded. Changing expansion toggles descendant visibility and updates the tree.



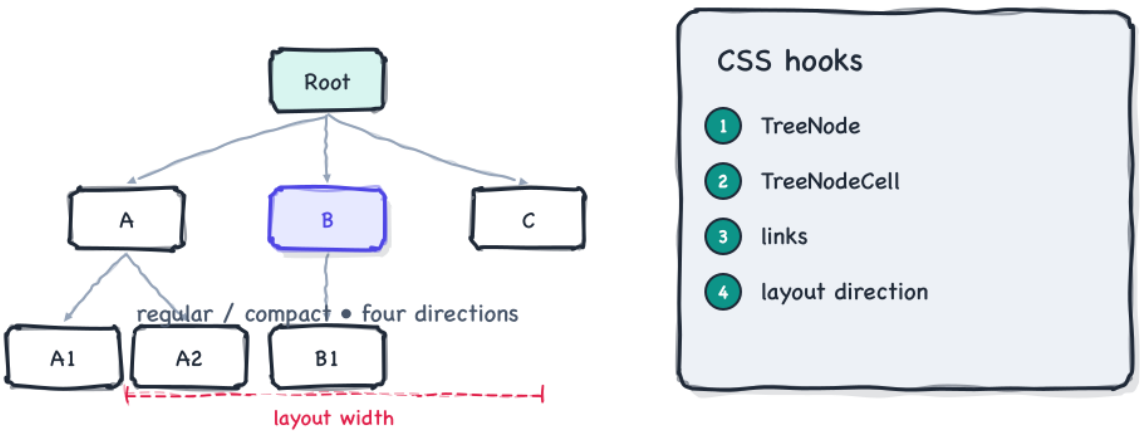
Data and interaction flow.

5.3 Links and style names

The skin asks linkStrategy to draw parent-child links and extra linkedNodes. If names exist, links receive classes such as link-parent-child or link-extra-source-target.

6. Styling

The style hooks below were verified in the control, skin and CSS sources.



Style hooks and visual states.

6.1 Style classes

Style class	Where used
tree-node-view	Root control style class.
tree-content	Group containing cells and links.
tree-node-cell	Default cell.

Style class	Where used
tree-node-cell-label	Label inside the default cell.
arrow-wrapper disclosure-arrow	Disclosure arrow shown when children exist.
link-arrow link-line link-path link-curve link-circle	Nodes returned by link strategies.
link-X-Y / link-extra-X-Y	Generated when TreeNode names are present.

6.2 Pseudo classes

Pseudo class	Meaning
ltr rtl ttb btt	Direction pseudo classes on TreeNodeView.
expanded collapsed	TreeNodeCell expansion state.

6.3 Styleable CSS properties

Property	Type	Default	Description
-fx-cell-width	Number	60	Default cell width.
-fx-cell-height	Number	30	Default cell height.
-fx-hgap	Number	20	Horizontal gap.
-fx-vgap	Number	50	Vertical gap.
-fx-node-line-gap	Number	10	Gap between node and link.
-fx-row-alignment	VPos	CENTER	TOP, CENTER, BOTTOM or BASELINE.
-fx-column-alignment	HPos	CENTER	LEFT, CENTER or RIGHT.
-fx-layout-type	LayoutType	REGULAR	REGULAR or COMPACT.
-fx-layout-direction	LayoutDirection	TOP_TO_BOTTOM	LEFT_TO_RIGHT, RIGHT_TO_LEFT, TOP_TO_BOTTOM or BOTTOM_TO_TOP.

```
.tree-node-view {
    -fx-layout-type: compact;
    -fx-layout-direction: left-to-right;
}
.tree-node-view .link-line {
    -fx-stroke: -fx-accent;
}
```

Example CSS.

7. Localization

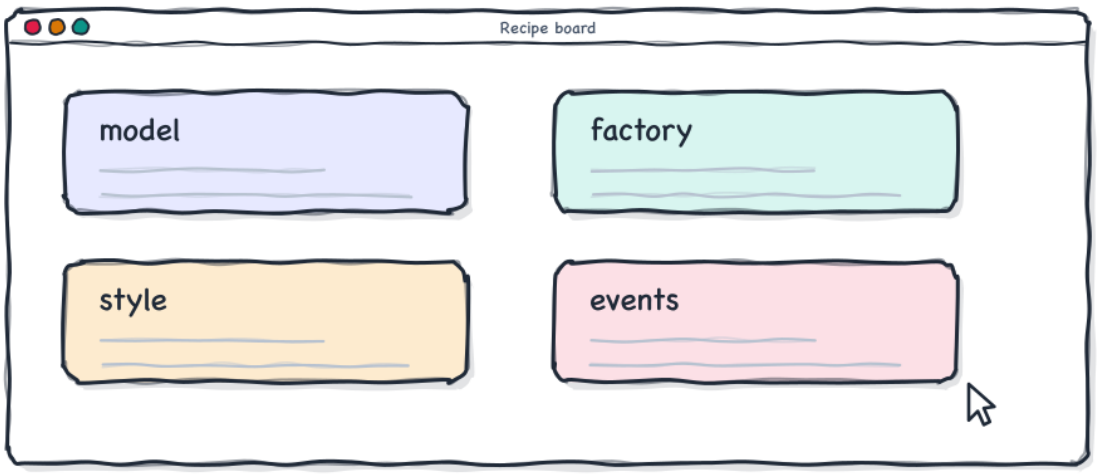
The following keys are read via `ResourceBundleManager`.

Key	English default
placeholder.no-root	No tree root.

8. Accessibility

TreeNodeView sets AccessibleRole.TREE_VIEW. The default TreeNodeCell does not set an explicit accessible role in its constructor.

9. Recipes



Common configuration recipes.

9.1 Use compact left-to-right layout

```
view.setLayoutType(TreeNodeView.LayoutType.COMPACT);
view.setLayoutDirection(TreeNodeView.LayoutDirection.LEFT_TO_RIGHT);
```

9.2 Per-node size

```
TreeNode<String> big = new TreeNode<>("Wide");
big.setSize(140, 50);
```

9.3 Add extra links

```
source.getLinkedNodes().add(target);
source.setName("source");
target.setName("target");
```

9.4 Custom cell factory

```
view.setCellFactory(value -> {
    TreeNodeCell<String> cell = new TreeNodeCell<>(value);
    cell.getStyleClass().add("org-node");
    return cell;
});
```

9.5 Refresh after model-side changes

```
view.refresh();
```

9.6 Checklist

1. Use TreeNode, not TreeItem.
2. Set TreeNode.name when you need stable link style classes.
3. Wrap the view in ScrollPane for large trees.
4. Choose rowAlignment only for vertical directions and columnAlignment for horizontal directions.

10. See also

Demo app: `TreeNodeViewApp`. Run it with:

```
mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.TreeNodeViewApp
```

- Related GemsFX controls: TreeNode, TreeNodeCell, link strategies, TreeView.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/