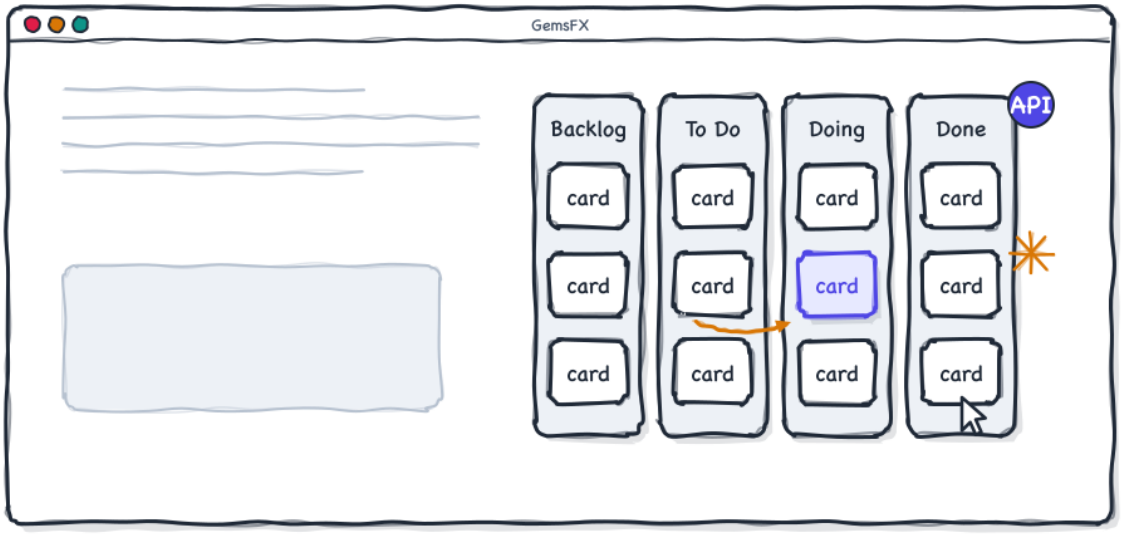
MultiColumnListView

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



Generated cartoon overview of MultiColumnListView.

MultiColumnListView displays several ListView columns with headers, separators, drag-and-drop markers, a placeholder and a LoadingPane overlay.

Table of Contents

1. Introduction

MultiColumnListView is a board-style control. Each `ListViewColumn` owns user items and internal `ColumnItem` wrappers so the control can insert From/To placeholders during drag and drop.

1.1 Key features

- Columns contain a header node, item list and userObject.
- Default list view factory creates AutoscrollListView instances.
- Drag and drop can be disabled or filtered through callbacks.
- Events report drag start, drag-over, veto and item moved steps.
- LoadingPane and placeholder support empty/loading states.

1.2 Maven dependency

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

Use package `com.dlsc.gemsfx`.

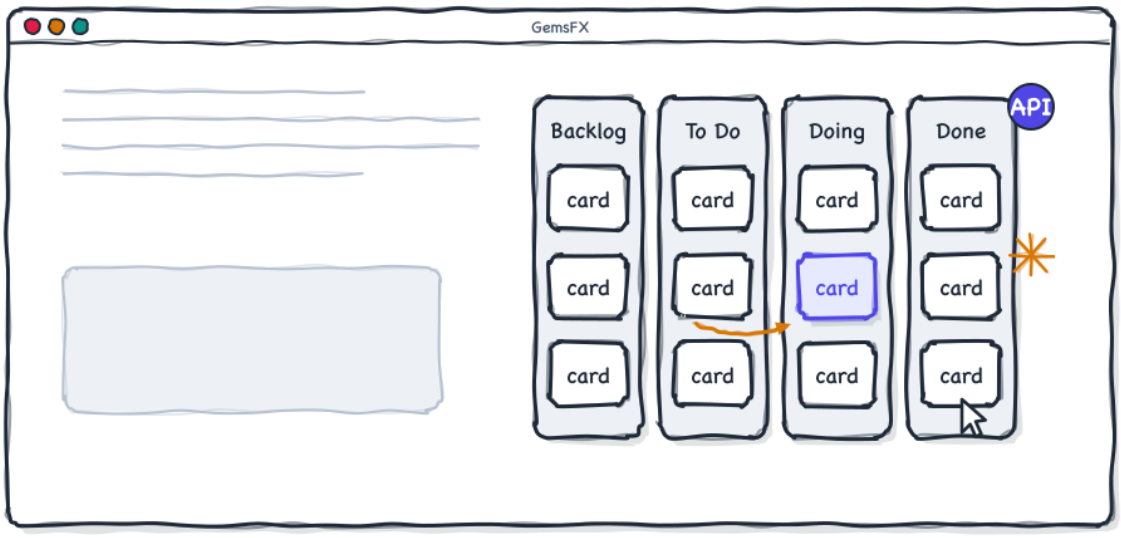
2. Getting started

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

```
MultiColumnListView<String> board = new MultiColumnListView<>();
ListViewColumn<String> todo = new ListViewColumn<>();
todo.setHeader(new Label("To Do"));
todo.getItems().setAll("Write docs", "Review PR");
ListViewColumn<String> done = new ListViewColumn<>();
done.setHeader(new Label("Done"));

board.getColumns().setAll(todo, done);
board.addEventHandler(MultiColumnListViewEvent.ITEM_MOVED, evt -> {
    System.out.println(evt.getDraggedItem() + " moved");
});
```

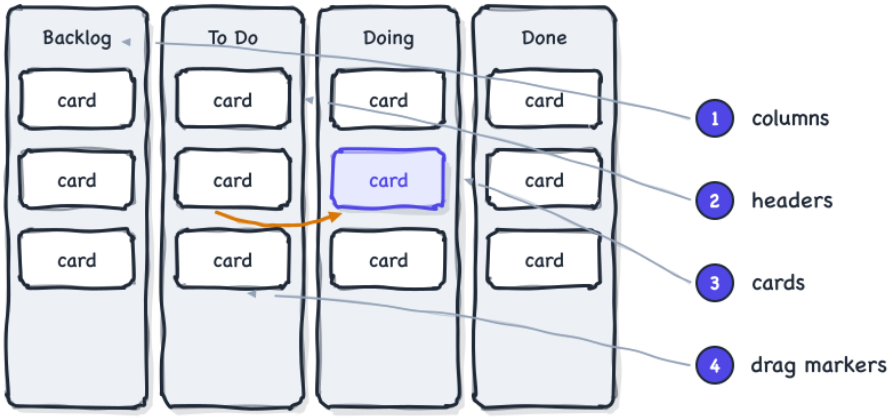
Minimal setup for `MultiColumnListView`.



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
MultiColumnListView	multi-column-list-view	Root control and API owner.
LoadingPane	loading-pane	Wraps the grid and shows loading/error states.
GridPane	grid-pane	One list column per ListViewColumn, plus optional separators.
ListViewColumn	model object	Header, user items, wrappers and userObject.

Part	Type / style	Description
ColumnListCell	column-list-cell	ListCell with drag-and-drop placeholder handling.

4. Control API

4.1 Columns and rendering

Property	Type	Default	Description
columns	ListProperty<ListViewColumn<T>>	empty list	Columns displayed by the view.
showHeaders	BooleanProperty	true	Shows each column header; styleable.
separatorFactory	ObjectProperty<Callback<Integer, Node>>	column-separator Region	Creates separator nodes between columns; null disables separators.
listViewFactory	ObjectProperty<Callback<MultiColumnListView<T>, ListView<ColumnItem<T>>>>	new Autoscroll ListView	Creates the ListView for each column.
cellFactory	ObjectProperty<Callback<MultiColumnListView<T>, ColumnListCell<T>>>	ColumnListCell::new	Creates list cells.

4.2 Drag and drop

Property	Type	Default	Description
disableDragAndDrop	BooleanProperty	false	Disables dragging; styleable.
dragPossibleCallback	ObjectProperty<Callback<T, Boolean>>	item -> true	Allows or rejects drag start for an item.
dropPossibleCallback	ObjectProperty<Callback<DropParameter<T>, Boolean>>	param -> true	Allows or rejects a drop target.
draggedItem	ObjectProperty<T>	null	Currently dragged user object.
draggedItems	ObservableList<T>	empty	Selected user objects involved in the drag.
fromPlaceholder / toPlaceholder	ColumnItem<T>	internal singletons	Marker wrappers inserted during drag operations.

4.3 Loading and placeholder

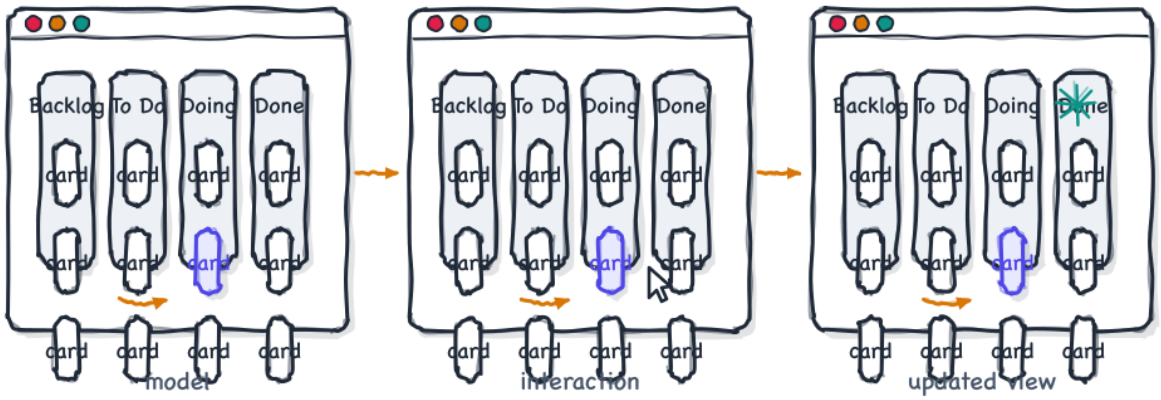
Property	Type	Default	Description
placeholder	ObjectProperty<Node>	Label "No columns defined."	Shown when there are no columns and loadingStatus is OK.
loadingStatus	ObjectProperty<LoadingPane.Status>	OK	Controls the LoadingPane.

Property	Type	Default	Description
loadingStatusSize	ObjectProperty<LoadingPane.Size>	MEDIUM	Loading indicator size.
progressIndicator	ObjectProperty<ProgressIndicator>	CircleProgressIndicator	Indicator used by LoadingPane.

5. Behaviour

5.1 Building columns

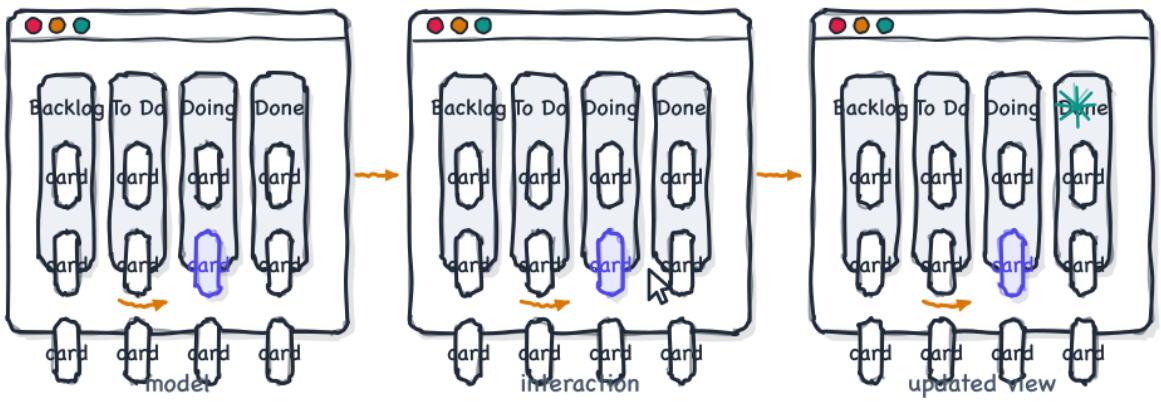
The skin creates equal-width GridPane columns, optional header row, ListView instances and optional separators. Empty columns get a placeholder label that accepts valid drops.



The main runtime behaviour.

5.2 Drag and drop markers

When a drag starts, the dragged wrapper is replaced by the from placeholder. While hovering, the to placeholder is inserted above or below the candidate cell.



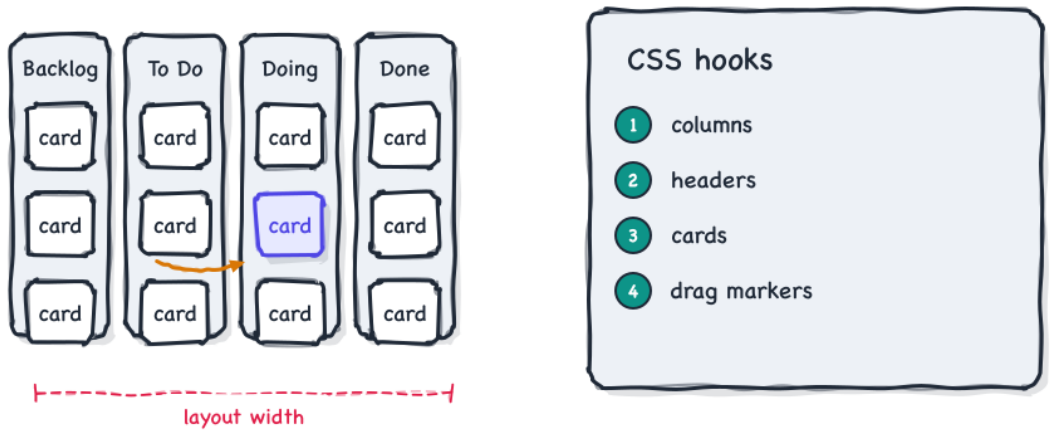
Data and interaction flow.

5.3 Events and vetoes

The control fires DRAG_STARTED, DRAG_NOT_POSSIBLE, DRAG_OVER, DROP_NOT_POSSIBLE and ITEM_MOVED with the dragged item, column and index.

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
multi-column-list-view	Root style class.
placeholder	Root placeholder and list placeholder.
loading-pane grid-pane	Loading wrapper and grid.
column-background-region column-foreground-region	Per-column overlay regions.
column-separator	Default separator node.
column-list-cell content-pane content-label	Default cell visuals.
from to	Pseudo states on ColumnListCell placeholders.
drag-over	Pseudo state on a column placeholder.

6.2 Pseudo classes

Pseudo class	Meaning
from	ColumnListCell shows the source marker.
to	ColumnListCell shows the target marker.

Pseudo class	Meaning
drag-over	ListView placeholder is a valid drag target.
hover	Column foreground follows ListView hover/drag hover.

6.3 Styleable CSS properties

Property	Type	Default	Description
-fx-show-headers	Boolean	true	Shows column headers.
-fx-disable-drag-and-drop	Boolean	false	Disables user drag-and-drop editing.

```
.multi-column-list-view {
    -fx-show-headers: true;
    -fx-disable-drag-and-drop: false;
}
.multi-column-list-view .column-list-cell:to > .content-pane {
    -fx-border-color: -fx-accent;
}
```

Example CSS.

7. Localization

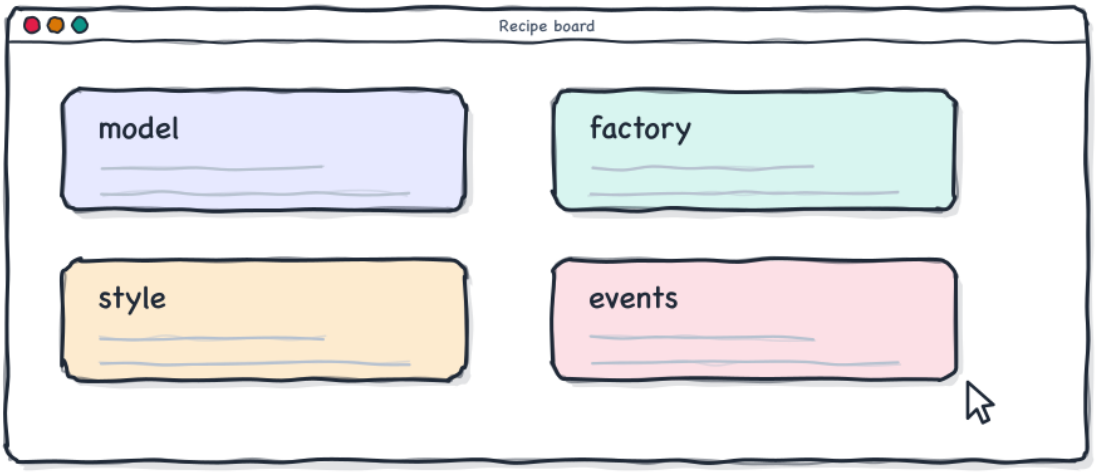
The following keys are read via `ResourceBundleManager`.

Key	English default
column.header.default	Column Header
placeholder.from	From
placeholder.to	To
placeholder.no-columns	No columns defined.

8. Accessibility

The constructor sets AccessibleRole.LIST_VIEW and sets focusTraversable to false.

9. Recipes



Common configuration recipes.

9.1 Disable editing

```
board.setDisableDragAndDrop(true);
```

9.2 Restrict drops

```
board.setDropPossibleCallback(param -> param.getColumn().getItems().size() < 10);
```

9.3 Custom list views

```
board.setListViewFactory(view -> new AutoscrollListView<>());
```

9.4 Hide headers

```
board.setShowHeaders(false);
```

9.5 Listen for moves

```
board.addEventHandler(MultiColumnListViewEvent.ITEM_MOVED, evt -> saveBoard());
```

9.6 Checklist

1. Use ListViewColumn.getItems(), not itemWrappers, for application data.
2. Override ColumnListCell.updateUserObject rather than updateItem.
3. Return false from callbacks to veto drags or drops.
4. The old PDF is intentionally replaced by this generated manual.

10. See also

Demo app: `MultiColumnListViewApp`. Run it with:

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

- Related GemsFX controls: AutoscrollListView, GridTableView, SelectionBox.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/