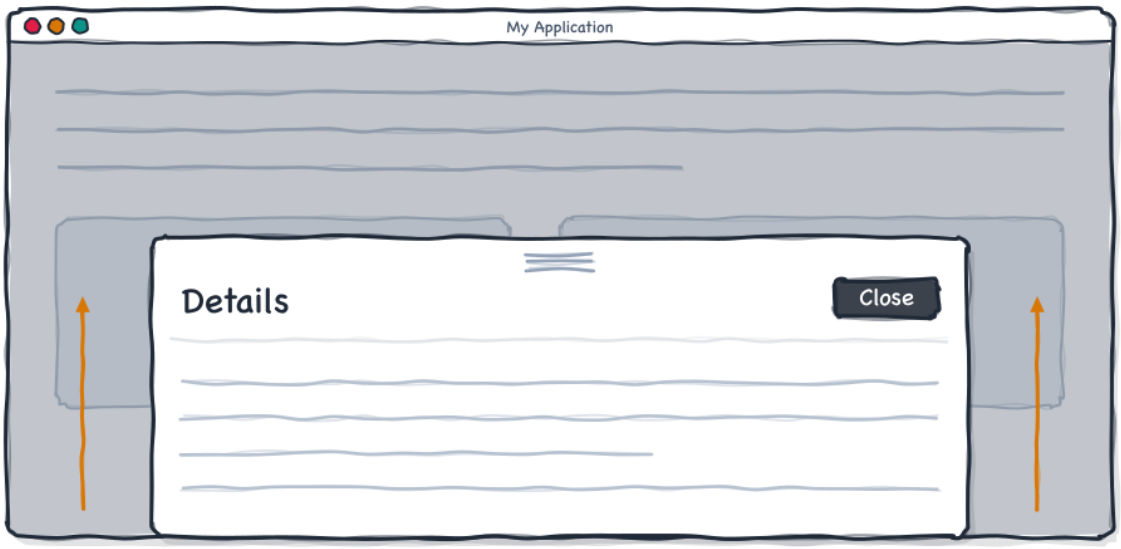
DrawerStackPane

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



The drawer slides in over a dimmed application window.

DrawerStackPane is a StackPane that can slide a resizable drawer in from the bottom edge. While the drawer is open the rest of the pane is covered by a semi-transparent glass pane that blocks user input. This manual explains the anatomy of the control, its API, its layout rules, its styling hooks and the most common usage patterns.

Table of Contents

1. Introduction

DrawerStackPane extends `javafx.scene.layout.StackPane` and adds a second, optional layer to it: a *drawer*. The drawer is a panel that slides in from the bottom edge of the pane, covering part of the regular content. It is a good fit for detail views, log output, filter panels, wizards or anything else that should temporarily take over the lower part of a window without opening a separate dialog.

The regular content is added the usual way, through the children list of the stack pane. The drawer receives its own content through the `drawerContent` property.

1.1 Key features

- The user can resize the drawer by dragging the header at its top.
- Dragging the header below the lower bounds of the pane closes the drawer completely.
- Opening and closing can be animated (see `animateDrawer`).
- While the drawer is open, a dark semi-transparent glass pane blocks input to the content below.
- The glass pane can fade in and out.
- The drawer can be given its own preferred width (see `preferredDrawerWidth`).
- The drawer height can be persisted automatically via the Java preferences API (see `preferencesKey`).
- Auto hiding: the drawer closes when the user clicks the background, and when ESCAPE is pressed.
- An optional title bar with a title, an extra node and a toolbar.

1.2 Maven dependency

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

The control lives in the module `com.dlsc.gemsfx`*, package* `com.dlsc.gemsfx`*.*

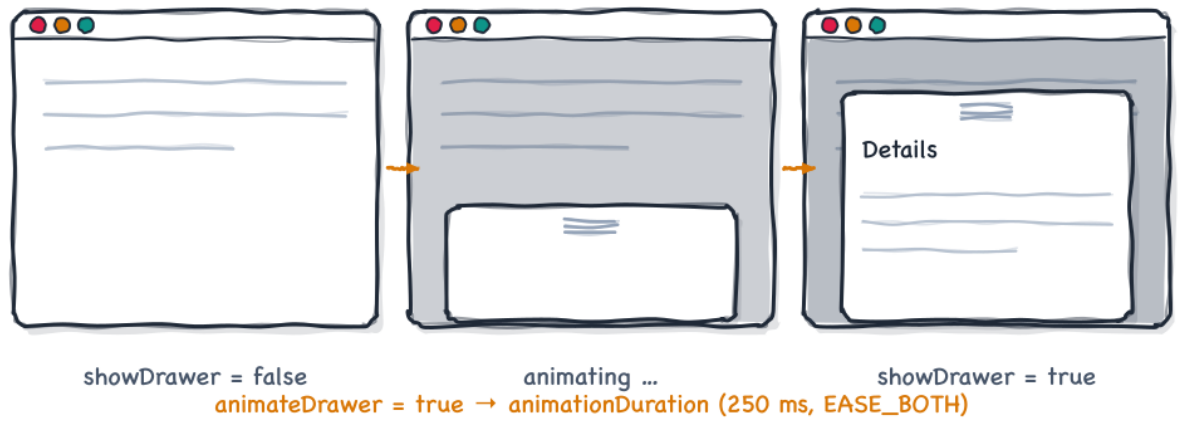
> **Note.** DrawerStackPane is a *layout*, not a `Control`. It has no skin class; the drawer is assembled directly by the pane and styled through its own user agent stylesheet `drawer-stackpane.css`.

2. Getting started

The minimal setup consists of three steps: create the pane, add the regular content, and set the content of the drawer. The drawer is then shown and hidden by toggling `showDrawer`.

```
DrawerStackPane pane = new DrawerStackPane();

// regular content: added like on any other stack pane
Button showButton = new Button("Show Drawer");
pane.getChildren().add(showButton);

// drawer content
Label details = new Label("Lorem ipsum dolor sit amet ...");
details.setWrapText(true);
pane.setDrawerContent(new ScrollPane(details));

// title bar (optional)
pane.setShowDrawerTitle(true);
pane.setDrawerTitle("Details");

// open the drawer
showButton.setOnAction(evt -> pane.setShowDrawer(true));
```

A complete, runnable setup of a DrawerStackPane.



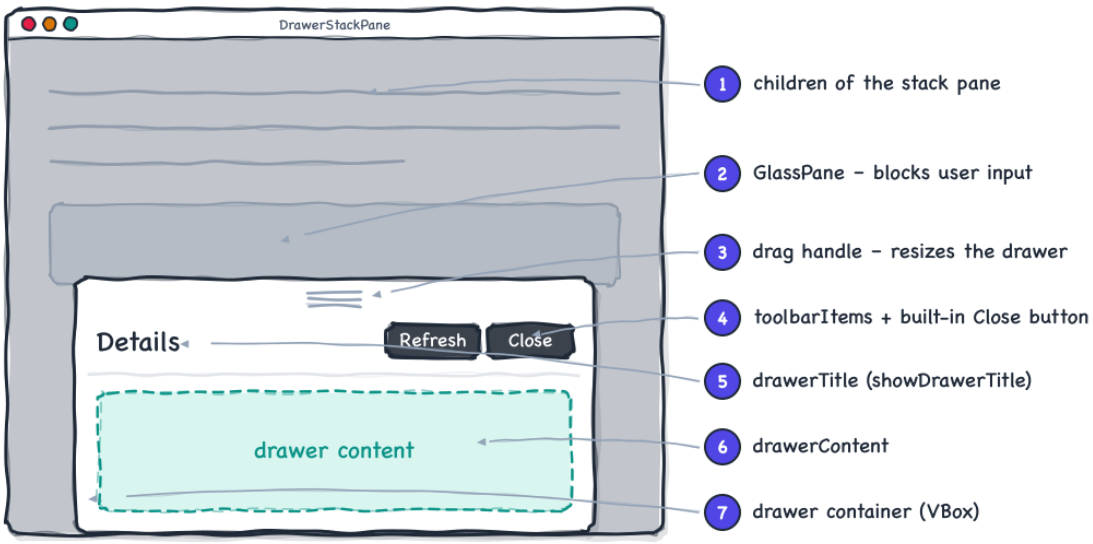
Closed, animating and open: setting `showDrawer` triggers the transition.

The drawer always spans the full remaining height of the pane below `topPadding`; how much of it is actually visible is controlled by `drawerHeight`, a value between `0` and `1`. Opening the drawer animates that value from `0` to the last height the user chose.

> **Tip.** A close button is added to the drawer toolbar automatically. You never have to build one yourself, and the drawer is always closable via ESCAPE.

3. Anatomy

The pane stacks three layers on top of each other: the regular children, the glass pane and the drawer. The drawer itself is a `VBox` with a header area on top and the application-provided content below.



The parts of a DrawerStackPane.

Part	Node	Description
Content	children of the pane	Everything added via `getChildren()`. Laid out by the standard stack pane algorithm.
Glass pane	`GlassPane`	Dark, semi-transparent overlay. Blocks mouse input to the content and closes the drawer when clicked (see `autoHide`).
Drawer	`VBox`	The sliding panel. Unmanaged and positioned manually by `layoutChildren()`.
Top container	`StackPane`	Stacks the header box and the drag handle on top of each other.
Drag handle	`StackPane` / `VBox`	Three short separators indicating that the drawer can be resized. Mouse transparent.
Header	`HBox`	Carries the title label, the optional extra node and the toolbar. This is the area the user drags.
Toolbar	`ToolBar`	Bound to `toolbarItems`; always contains the built-in close button.
Drawer content	any `Node`	Set via `drawerContent`. Gets `Priority.ALWAYS` for vertical growth.

4. Control API

4.1 Content and visibility

Property	Type	Default	Description
drawerContent	ObjectProperty<Node>	null	The node shown inside the drawer. Replacing it removes the previous node from the drawer.
showDrawer	BooleanProperty	false	Opens (`true`) or closes (`false`) the drawer, with animation if enabled.
autoHide	BooleanProperty	true	When true, a primary-button click on the glass pane triggers `onCloseRequest`.

4.2 Title bar and toolbar

Property	Type	Default	Description
showDrawerTitle	BooleanProperty	false	Shows the title label. When false the header content is aligned to the right instead of the left. Styleable.
drawerTitle	StringProperty	"Untitled"	The text of the title label. The default value is localized.
drawerTitleExtra	ObjectProperty<Node>	null	An additional node appended to the header box, for example a status indicator.
toolbarItems	ListProperty<Node>	[close button]	The items of the drawer toolbar. The built-in close button is already part of this list.

4.3 Animation

Property	Type	Default	Description
animateDrawer	BooleanProperty	true	Animates opening and closing. Also drives the fading of the glass pane. Styleable.
animationDuration	ObjectProperty<Duration>	250 ms	Duration of the slide animation and of the glass pane fade. Styleable.
fadeInOut	BooleanProperty	true	Whether the glass pane appears and disappears smoothly. Styleable.

4.4 Callbacks

Closing the drawer is routed through a single callback. This makes it easy to ask the user for confirmation, or to close the drawer from somewhere else in the application.

Property	Type	Default	Description
onCloseRequest	ObjectProperty<Runnable>	sets showDrawer to false	Invoked by the close button, by a click on the glass pane, by ESCAPE and when the drawer is dragged out of view.
onDrawerClose	ObjectProperty<Runnable>	null	Invoked *after* the drawer has become invisible, so after the closing animation has finished.

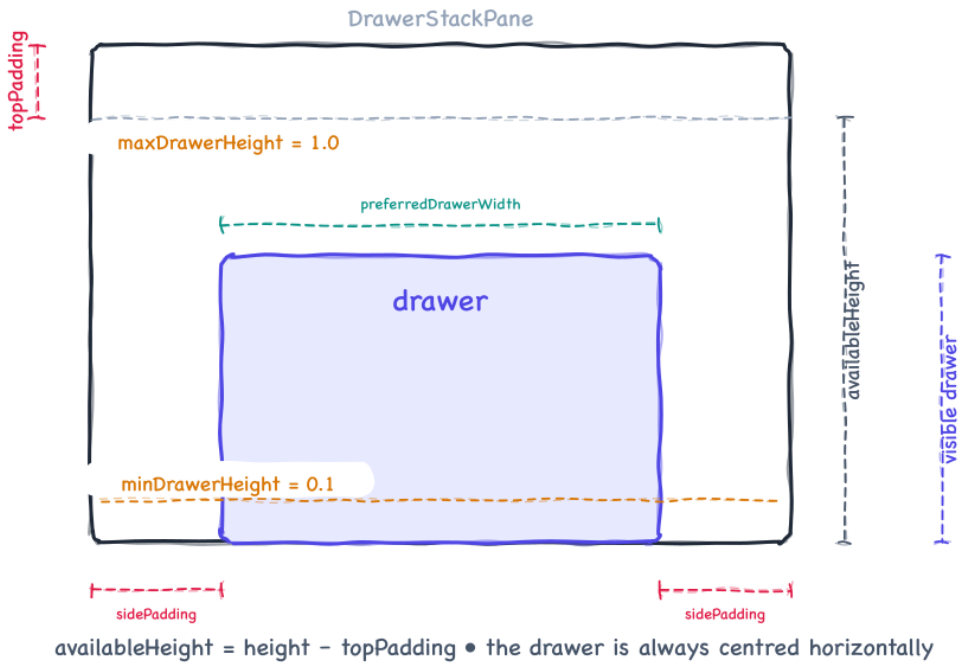
```
pane.setOnCloseRequest(() -> {
    if (editor.isDirty()) {
        showConfirmationDialog();
    } else {
        pane.setShowDrawer(false);
    }
});

pane.setOnDrawerClose(() -> editor.reset());
```

Intercepting the close request and reacting to the finished close.

> **Careful.** If you replace `onCloseRequest`, make sure some code path still calls `setShowDrawer(false)` - otherwise the drawer can no longer be closed by the user.

5. Layout and sizing

The drawer is not managed by the stack pane layout. Instead `layoutChildren()` positions it explicitly, horizontally centred and anchored to the bottom edge.



How the drawer bounds are derived from the properties of the pane.

```
availableHeight = getHeight() - getTopPadding()
maxDrawerWidth  = getWidth() - 2 * getSidePadding()

x = (getWidth() - drawerWidth) / 2
y = getHeight() - getDrawerHeight() * availableHeight

drawer.resizeRelocate(x, y, drawerWidth, availableHeight * getDrawerHeight())
```

The layout algorithm in pseudo code.

5.1 Sizing properties

Property	Type	Default	Description
drawerHeight	DoubleProperty	0.7 (persisted)	The visible fraction of the available height. Animated while opening and closing, and changed by dragging.
minDrawerHeight	DoubleProperty	0.1	Lower bound for dragging, as a fraction. Must not be negative. Styleable.
maxDrawerHeight	DoubleProperty	1.0	Upper bound for dragging, as a fraction. Must not be greater than 1. Styleable.
preferredDrawerWidth	DoubleProperty	Double.MAX_VALUE	Width of the drawer in pixels. The default makes the drawer as wide as the pane minus the side paddings. Styleable.
topPadding	DoubleProperty	20	Space kept free above a fully opened drawer, in pixels. Must not be negative. Styleable.

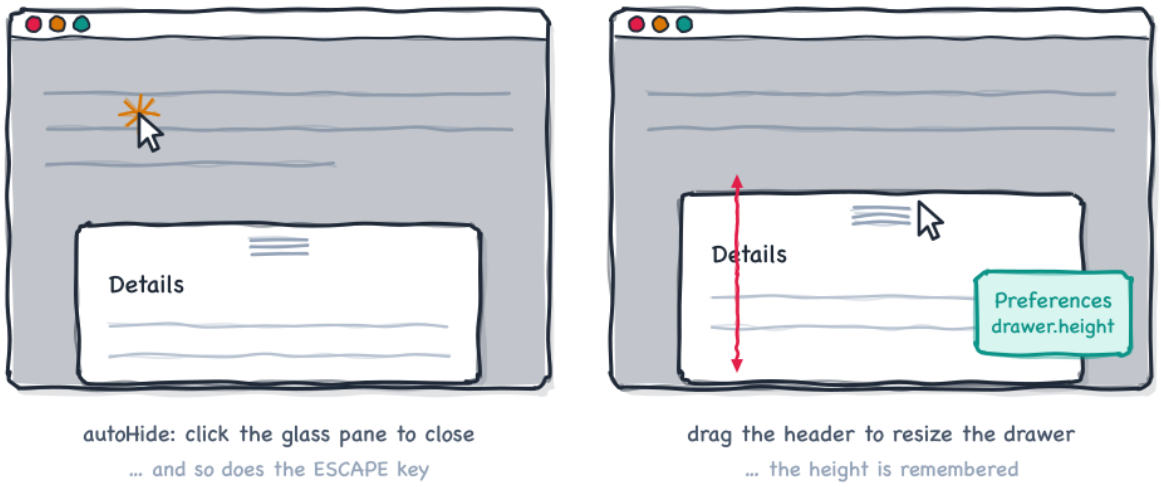
Property	Type	Default	Description
sidePadding	DoubleProperty	100	Space kept free left and right of the drawer, in pixels. Must not be negative. Styleable.

Setting `preferredDrawerWidth` to `Region.USE_PREF_SIZE` makes the drawer exactly as wide as its own preferred width, which is useful for narrow forms. Any other value is capped at the width of the pane minus twice the side padding.

```
pane.setPreferredDrawerWidth(Region.USE_PREF_SIZE); // as wide as the content needs
pane.setPreferredDrawerWidth(600);                  // 600 pixels, capped by sidePadding
pane.setSidePadding(0);                             // full width drawer
pane.setTopPadding(60);                             // never cover the toolbar
```

Careful. `minDrawerHeight`, `maxDrawerHeight`, `topPadding` and `sidePadding` validate their values and throw an `IllegalArgumentException` for values outside their legal range.

6. Interaction and persistence



Closing by clicking the glass pane, and resizing by dragging the header.

6.1 Gestures and keys

Gesture	Effect
Drag the header up or down	Resizes the drawer within the min / max bounds.
Drag the header below the bottom edge	Triggers `onCloseRequest`, so the drawer closes.
Double click the header	Maximises the drawer, or closes it when it is already maximised.
Primary click on the glass pane	Triggers `onCloseRequest` if `autoHide` is true.
ESCAPE	Triggers `onCloseRequest` while the drawer is open.
Close button in the toolbar	Triggers `onCloseRequest`.

6.2 Persisting the drawer height

Whenever the user finishes a drag, the current `drawerHeight` is written to the Java preferences of the package of `DrawerStackPane`, under the key `<preferencesKey>.drawer.height`. The next time the drawer opens, that value is restored and clamped to the range `[0.1, 1.0]`.

Property	Type	Default	Description
preferencesKey	StringProperty	"drawer.stackpane"	Prefix of the preferences key. Set it to null to disable persistence, or to a unique value if an application uses several drawers.

```
// give every drawer of the application its own remembered height
pane.setPreferencesKey("customer.details");

// or opt out of persistence completely
pane.setPreferencesKey(null);
```

Note. If the drawer height was never persisted, the drawer opens at `0.7` of the available height.

7. Styling

The control brings its own user agent stylesheet, `drawer-stackpane.css`. All style classes below are nested inside the root class `.drawer-stackpane`.

7.1 Style classes

Selector	Node
`.drawer-stackpane`	the pane itself
`> .drawer`	the sliding drawer container
`> .drawer > .top`	container of header and drag handle
`> .top > .header`	the draggable header box
`> .header > .title-label`	the drawer title
`> .header > .tool-bar`	the toolbar of the drawer
`> .tool-bar > .container > .button`	toolbar buttons, including the close button
`.close-button`	the built-in close button
`> .top > .drag-handle`	the drag handle area
`> .drag-handle > .handle`	the three separator lines
`.glass-pane`	the input-blocking overlay

7.2 Styleable CSS properties

CSS property	Type	Default
`-fx-animate-drawer`	boolean	true
`-fx-animation-duration`	Duration	250ms

CSS property	Type	Default
`-fx-drawer-side-padding`	number	100
`-fx-drawer-top-padding`	number	20
`-fx-fade-in-out`	boolean	true
`-fx-max-drawer-height`	number	1.0
`-fx-min-drawer-height`	number	0.1
`-fx-preferred-drawer-width`	number	Double.MAX_VALUE
`-fx-show-drawer-title`	boolean	false

```
.drawer-stackpane {
    -fx-drawer-side-padding: 40;
    -fx-drawer-top-padding: 60;
    -fx-min-drawer-height: 0.25;
    -fx-animation-duration: 400ms;
    -fx-show-drawer-title: true;
}

.drawer-stackpane > .drawer {
    -fx-background-color: -fx-box-border, -fx-background;
    -fx-background-radius: 8 8 0 0;
}
```

Styling the drawer through CSS instead of Java code.

> **Note.** Because these are styleable properties, values set from Java code win over values coming from a stylesheet unless the stylesheet uses `!important`.

8. Localization

The two strings of the control, the label of the close button and the default drawer title, are loaded through `ResourceBundleManager` from the bundle `drawer-stack-pane.properties`.

Key	English text
`button.close`	Close
`title.untitled`	Untitled

Translations ship for Arabic, Chinese, Danish, Dutch, Finnish, French, German, Italian, Japanese, Norwegian, Portuguese, Spanish, Swedish and Ukrainian. To override a text, put a bundle with the same base name earlier on the class path, or simply set `drawerTitle` yourself.

9. Recipes

9.1 A drawer without a title bar

Leaving `showDrawerTitle` at its default hides the title label and aligns the toolbar to the right edge of the header.

```
pane.setShowDrawerTitle(false); // default
```

9.2 Custom toolbar items

Items are added to `toolbarItems`. Because the built-in close button is already part of that list, add your own items at the front to keep the close button on the right.

```
Button refresh = new Button("Refresh");
pane.getToolbarItems().add(0, refresh);
```

9.3 An extra node in the header

```
ProgressIndicator indicator = new ProgressIndicator();
indicator.setPrefSize(16, 16);
pane.setDrawerTitleExtra(indicator);
```

9.4 Opening the drawer without animation

```
pane.setAnimateDrawer(false);
pane.setShowDrawer(true);
```

9.5 Checklist

1. Add the regular content via `getChildren()`, not via `drawerContent`.
2. Give every drawer of the application its own `preferencesKey`.
3. Wrap long drawer content in a `ScrollPane`.
4. Keep `topPadding` large enough that the user can still see where the drawer came from.
5. If you override `onCloseRequest`, keep a path that closes the drawer.

10. See also

- Demo application: `com.dlsc.gemsfx.demo.DrawerStackPaneApp` (run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.DrawerStackPaneApp`)
- `GlassPane` - the overlay used by this control, also usable standalone.
- `HiddenSidesPane` - for panels sliding in from any of the four sides.
- `DialogPane` - when a modal dialog is the better fit.
- API documentation: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/