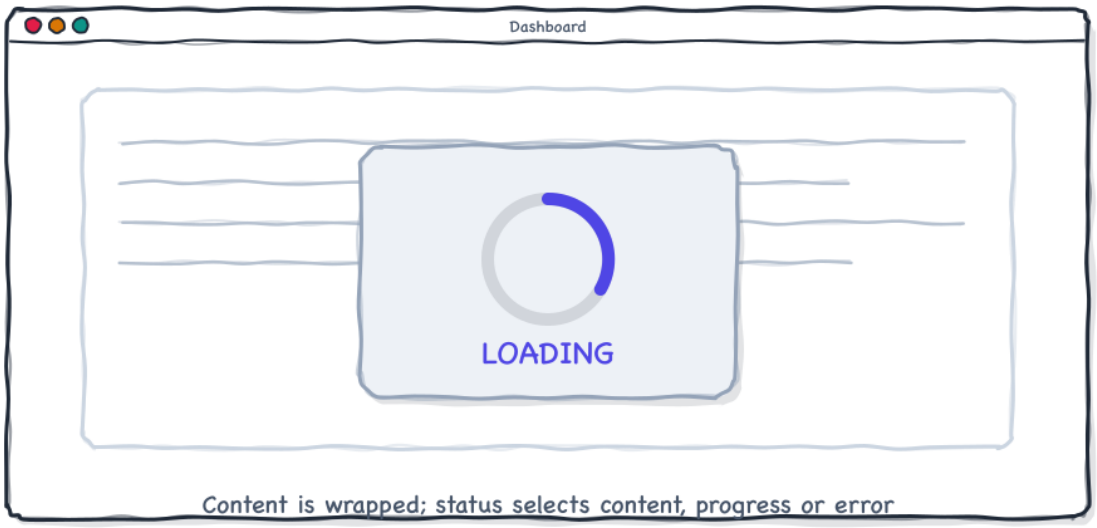
LoadingPane

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



LoadingPane replaces wrapped content with progress feedback or an error message.

LoadingPane wraps a content node and switches between normal content, progress feedback and an error node. It delays committing LOADING to avoid flicker, binds its progress to the active progress indicator and exposes CSS hooks for indicator size.

Table of Contents

1. Introduction

LoadingPane extends `StackPane` and is annotated with `@DefaultProperty("content")`. It is useful around tables, lists, forms or dashboards that need to show loading and failure states without rebuilding the surrounding scene graph.

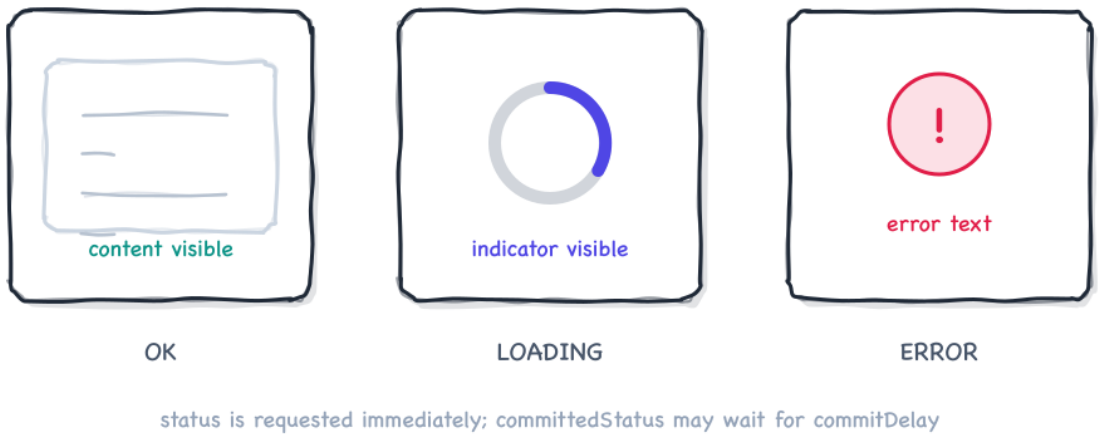
- Status `OK` shows the content node.
- Status `LOADING` shows a progress indicator wrapper.
- Status `ERROR` shows an error node.
- Progress value `1.0` automatically returns the pane to `OK`.
- Convenience methods `load()`, `ok()` and `error(...)` are thread-safe via `Platform.runLater`.

```
Label content = new Label("Some content goes here...");
LoadingPane pane = new LoadingPane(content);
pane.load();
```

2. Getting started

Create the pane with a content node or set `content` later. The default progress indicator is a `CircleProgressIndicator` with an extra `busy-indicator` style class.

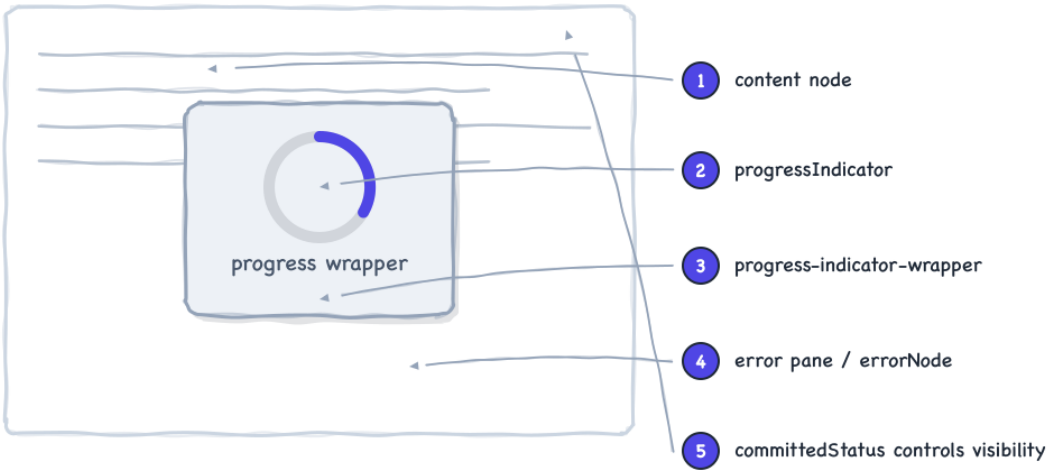
```
LoadingPane pane = new LoadingPane(tableView);
pane.setCommitDelay(200);

service.setOnRunning(evt -> pane.load());
service.setOnSucceeded(evt -> pane.ok());
service.setOnFailed(evt -> pane.error(service.getException()));
pane.progressProperty().bind(service.progressProperty());
```



The committed status decides which child is visible and managed.

Note. Only the transition to LOADING is delayed. OK and ERROR commit without the loading delay.

3. Anatomy



The child layers maintained by LoadingPane.

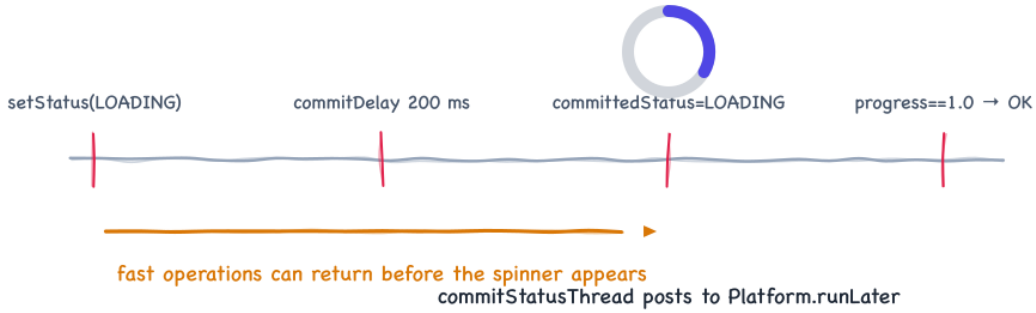
Part	Style class	Description
Root	`loading-pane`	The StackPane itself.
Content	application node	Visible only when committedStatus is OK.
Indicator wrapper	`progress-indicator-wrapper`	StackPane around the progress indicator; its size is styled by pseudo-class.
Progress indicator	`busy-indicator`	Default is CircleProgressIndicator; replaceable.
Error pane	`error-pane`	Wrapper around errorNode.
Error label	`error-label`	Default label with an icon and text bound to error.

4. Control API

Property	Type	Default	Description
content	ObjectProperty<Node>	null	Wrapped application content. Its visible property is bound to committedStatus == OK.
status	ObjectProperty<Status>	OK	Requested state: OK, LOADING or ERROR.
committedStatus	ReadOnlyObjectProperty<Status>	OK	Actual state after applying commitDelay for LOADING.
progress	DoubleProperty	0	Progress value bound to the active progress indicator. Setting 1.0 changes status to OK.
commitDelay	LongProperty	200	Delay in milliseconds before committing LOADING. Negative values throw IllegalArgumentException.
error	StringProperty	null	Text shown by the default error label.

Property	Type	Default	Description
errorNode	ObjectProperty<Node>	default Label	Node shown for ERROR.
progressIndicator	ObjectProperty<ProgressIndicator>	CircleProgressIndicator	Indicator used in LOADING and bound to progress.
size	ObjectProperty<Size>	MEDIUM	Styleable enum SMALL, MEDIUM or LARGE; reflected as pseudo-classes.

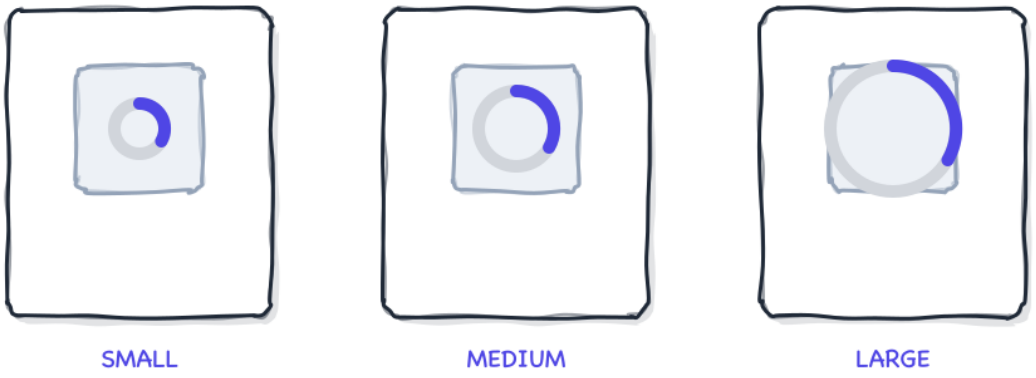
5. Behaviour and state flow



The delayed commit avoids short-lived loading flicker.

Action	Effect
`load()`	Requests LOADING and clears error text.
`ok()`	Requests OK and clears error text.
`error(String)`	Requests ERROR and stores the message.
`error(Throwable)`	Uses throwable message.
`setProgress(1.0)`	Sets status to OK.

6. Layout and sizing



The Size enum maps to wrapper dimensions in CSS.

The control rebuilds its child list when content, errorNode or progressIndicator changes. The progress indicator is placed inside an unmanaged-size wrapper whose preferred size comes from CSS. The indicator pref width and height are bound to the wrapper content area after padding.

Size	Wrapper size from CSS
SMALL	60px
MEDIUM	90px
LARGE	150px

7. Styling with CSS

The user agent stylesheet is `loading-pane.css`.

Selector	Purpose
`.loading-pane`	Root StackPane.
`.progress-indicator-wrapper`	Wrapper around the current ProgressIndicator; default padding 20px and size 90px.
`:small`	Wrapper size 60px.
`:large`	Wrapper size 150px.
`.error-pane > .error-label`	Red centered wrapping error text.
`.error-label .icon`	Built-in red error icon shape.

CSS property	Type	Default
`-fx-size`	Size enum	MEDIUM

```
.loading-pane {
    -fx-size: large;
}

.loading-pane > .error-pane > .error-label {
    -fx-text-fill: -fx-accent;
}
```

8. Recipes

8.1 Use a different indicator

```
RingProgressIndicator ring = new RingProgressIndicator();
pane.setProgressIndicator(ring);
pane.progressProperty().bind(service.progressProperty());
```

8.2 Replace the error node

```
Label error = new Label();
error.textProperty().bind(pane.errorProperty());
error.getStyleClass().add("my-error");
pane.setErrorNode(error);
```

8.3 Checklist

1. Keep content visibility under LoadingPane control; it binds the visible property.
2. Use commitDelay to suppress flicker for fast operations.
3. Set progress to values below zero for indeterminate work.
4. Call ok() or error(...) from background callbacks; the methods marshal to the FX thread.

9. Troubleshooting

- A negative commitDelay throws an IllegalArgumentException.
- If the spinner is too large or small, set `size` or style `-fx-size`.
- If replacing the progress indicator, use a ProgressIndicator subclass so progress binding exists.
- The class does not set a custom AccessibleRole in the source.