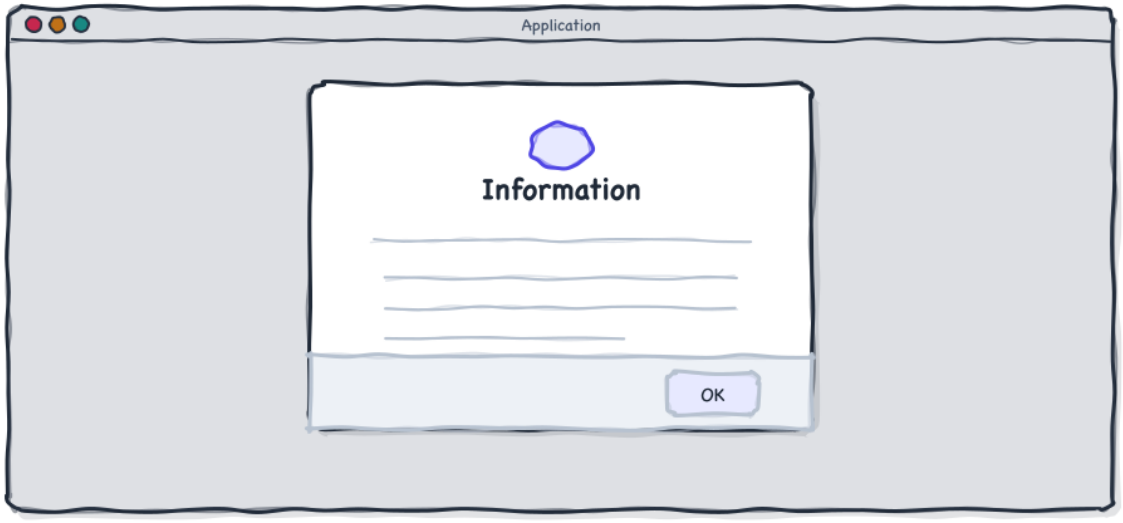
DialogPane

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



DialogPane shows typed dialogs above application content while a glass pane blocks the background.

DialogPane is a StackPane overlay for lightweight application dialogs. It manages a glass pane, one or more Dialog models, typed helper methods, custom content, validation, resizing, button bars, blocking and animation.

Table of Contents

1. Introduction

DialogPane is a StackPane that should cover the application area. It stays unmanaged and invisible while no dialogs are active, then displays a glass pane and one or more animated dialog content panes.

1.1 Key features

- Typed helpers for information, warning, confirmation, error, text input, custom node and busy dialogs.
- Dialog models with title, content, value, validation, button types and close callbacks.
- Global glass pane plus nested blocking glass panes for older dialogs.
- Default header and footer factories, both replaceable.
- OS-specific ButtonBar ordering and button validation rules.
- Resizable dialogs with optional Preferences persistence.
- ESC cancels the top dialog; SHIFT+ESC hides all dialogs.

1.2 Maven dependency

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

GemsFX controls live in module `com.dlsc.gemsfx`.

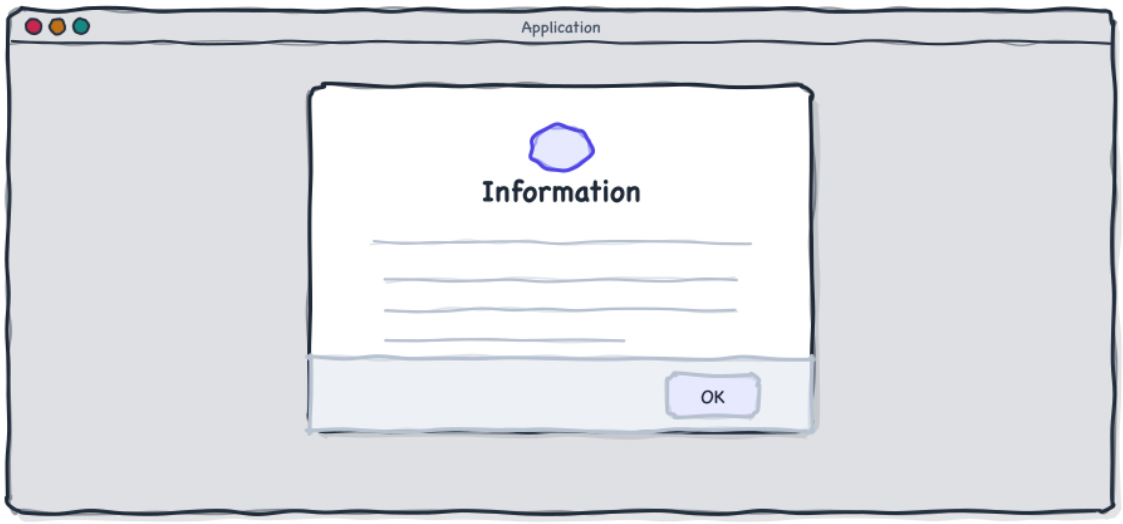
2. Getting started

```
StackPane root = new StackPane(applicationContent);
DialogPane dialogs = new DialogPane();
root.getChildren().add(dialogs);

dialogs.showInformation("Saved", "The document has been saved.");

dialogs.showConfirmation("Delete", "Really delete this item?")
       .onClose(button -> {
           if (button == ButtonType.YES) {
               deleteItem();
           }
       });
```

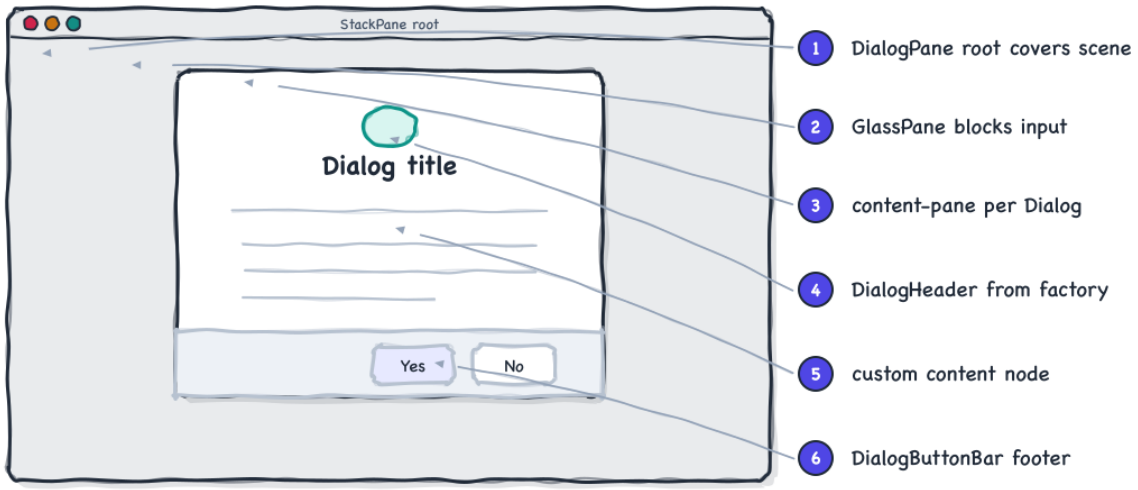
Create the pane as the last child of a StackPane and use helper methods.



A dialog is centered over the application while the glass pane blocks input.

3. Anatomy

Each active Dialog is represented by a ContentPane. The newest dialog is interactive; older dialogs receive a nested glass pane and become blocked.



The layers and default parts of DialogPane.

Part	Style class	Description
.dialog-pane	Root StackPane; transparent, fills available area.	
.glass-pane	Global glass pane over application content.	
.content-pane	Per-dialog shell; receives type style classes.	

Part	Style class	Description
.information, .warning, .error, .confirmation, .input, .blank	Dialog type classes.	
.vbox	Header/content/footer column.	
.header, .title-and-icon-box, .title, .icon, .close-button	Default DialogHeader parts.	
.content, .padding, .message-label, .prompt-node-wrapper	Content wrappers and message labels.	
.button-bar, .footer	Default DialogButtonBar footer.	

4. Pane API

4.1 Factories, dialogs and animation

Property	Type	Default	Description
headerFactory	ObjectProperty<Callback<Dialog<?>, Node>>	DialogHeader::new	Factory for the default title and close-button area.
footerFactory	ObjectProperty<Callback<Dialog<?>, Node>>	DialogButtonBar::new	Factory for the default ButtonBar footer.
dialogs	ListProperty<Dialog<?>>	empty list	Currently active dialogs. Adding and removing drives animations.
animationDuration	ObjectProperty<Duration>	100 ms	Duration for dialog fly-in/fly-out and glass pane fade.
animateDialogs	BooleanProperty	true	Enables the fly-in/fly-out transition.
fadeInOut	BooleanProperty	true	Animates dialog and glass-pane opacity.
showingDialog	ReadOnlyBooleanProperty	false	True while dialogs is not empty.
maximizedPadding	DoubleProperty	20	Insets kept around maximized dialogs.
labelSupplier	ObjectProperty<Supplier<Label>>	Label::new	Supplier for message labels in standard dialogs.
sendButtonText	StringProperty	Send	Text for the optional send button in detailed error dialogs.
converter	ObjectProperty<StringConverter<ButtonType>>	buttonType.getText()	Converts ButtonType text for footer buttons.

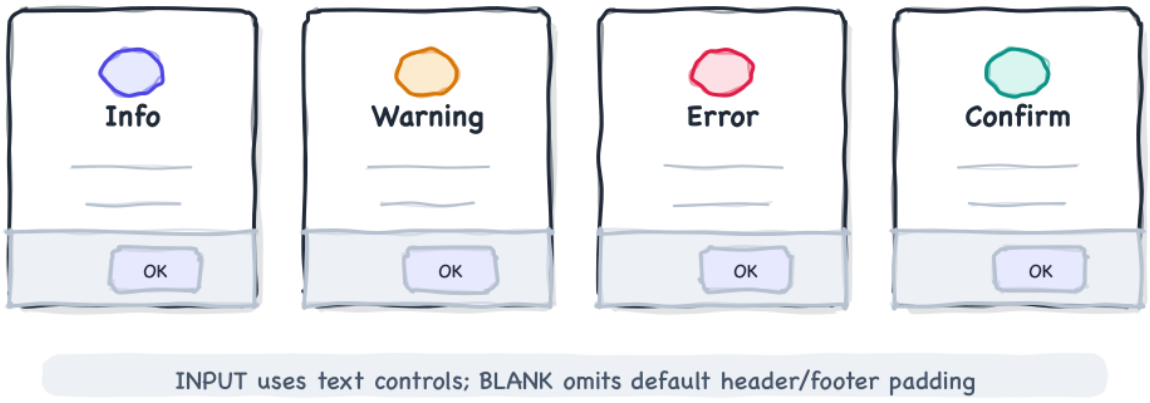
5. Dialog model API

Property	Type	Default	Description
id	StringProperty	null	Optional application identifier.
preferences	ObjectProperty<Preferences>	null	Stores/restores width and height through DefaultResizeHandler.
pref/min/max width and height	DoubleProperty	USE_COMPUTED_SIZE	Size constraints mirrored to the ContentPane.
required	BooleanProperty	false	Adds required-text validation for text input dialogs.
showCloseButton	BooleanProperty	true	Observed by the default header close button.
delay	ObjectProperty<Duration>	null	Optional delay before the dialog is added to the pane.
usingPadding	BooleanProperty	true; BLANK sets false	Adds the padding style class around content.
contentAlignment	ObjectProperty<Pos>	CENTER_LEFT	Alignment of the content node in the content area.
value	ObjectProperty<T>	null	Value produced by input/custom dialogs.
valid	BooleanProperty	true	Disables validating default buttons when false.

Property	Type	Default	Description
buttonTypes	ObservableList<ButtonType>	depends on Type	Footer button definitions.
sameWidthButtons	BooleanProperty	true	Uses ButtonBar uniform-size buttons.
maximize	BooleanProperty	false	Uses all available pane area minus maximizedPadding.
content	ObjectProperty<Node>	null	Main content node.
extras	ObjectProperty<Node>	null	Optional node for custom header/footer factories.
title	StringProperty	Dialog	Title used by DialogHeader.
showHeader / showFooter	BooleanProperty	true / true	Controls default header and footer visibility.
validator	ObjectProperty<Validator>	new Validator()	ValidatorFX validator run by committing buttons.
resizable	BooleanProperty	false	Installs ResizingBehaviour when true.
onResize	ObjectProperty<BiConsumer>	DefaultResizeHandler	Receives new size; default persists to Preferences.

6. Dialog types and helper methods

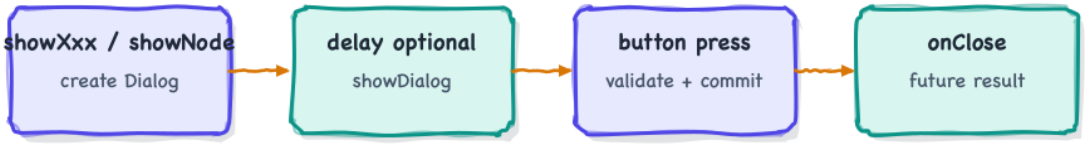
The Type enum contains INPUT, INFORMATION, ERROR, WARNING, CONFIRMATION and BLANK. The constructor selects default buttons: input and warning use OK/CANCEL, information and error use OK, confirmation uses YES/NO, and blank dialogs have no standard padding.



Built-in dialog types and their default presentation.

- showError supports message-only, exception-details, text details and optional send action.
- showTextInput chooses TextField or ResizableTextArea and binds dialog.value to text.
- showNode accepts any Node, maximize, custom buttons, same-width buttons and an optional validity property.
- showBusyIndicator creates a delayed blank busy dialog with no buttons.

7. Lifecycle, buttons and validation



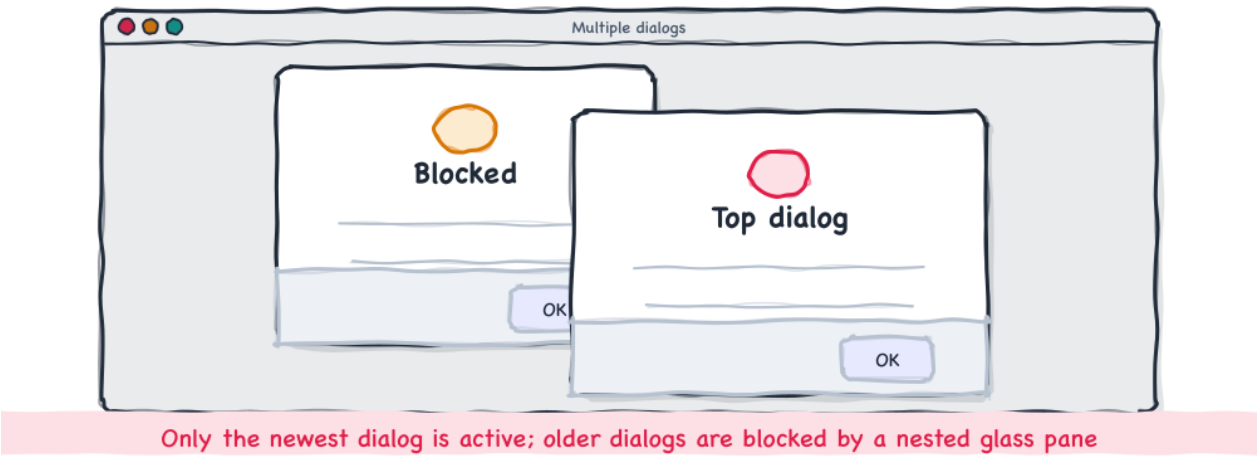
Dialog lifecycle from helper method to close callback.

Calling `show()` or a helper adds the dialog immediately, unless delay is set. Pressing a validating button runs the ValidatorFX validator; invalid input keeps the dialog open. CANCEL calls cancel(), clears value and

commits ButtonType.CANCEL.

> **Note.** The current source exposes `Dialog<T>` and `onClose` callbacks. It does not expose a Future or CompletionStage result API.

8. Blocking, keyboard and focus



Multiple dialogs stack; only the top dialog is active.

- ESC cancels the most recently opened dialog.
- SHIFT+ESC calls hideAllDialogs().
- Input dialogs attempt to keep focus inside the dialog content.
- A double-click in ListView content fires the default button when one exists.

9. Customization recipes

9.1 Custom content

```
DialogPane.Dialog<Object> dialog = dialogs.showNode(DialogPane.Type.INFORMATION,
        "Select item", tableView, true, List.of(ButtonType.OK, ButtonType.CANCEL));
dialog.setResizable(true);
```

9.2 Persist size

```
dialog.setId("preferences.dialog");
dialog.setPreferences(Preferences.userNodeForPackage(App.class).node(dialog.getId()));
dialog.setResizable(true);
```

9.3 Uppercase buttons

```
dialogs.setConverter(new StringConverter<>() {
    public String toString(ButtonType type) { return type == null ? "" : type.getText().toUp
perCase(); }
    public ButtonType fromString(String text) { return null; }
});
```

10. Styling

Style class	Description
.dialog-pane	Root StackPane; transparent, fills available area.
.glass-pane	Global glass pane over application content.
.content-pane	Per-dialog shell; receives type style classes.
.information, .warning, .error, .confirmation, .input, .blank	Dialog type classes.
.vbox	Header/content/footer column.
.header, .title-and-icon-box, .title, .icon, .close-button	Default DialogHeader parts.
.content, .padding, .message-label, .prompt-node-wrapper	Content wrappers and message labels.
.button-bar, .footer	Default DialogButtonBar footer.
.error-container, .error-text-area	Detailed error-dialog content.
.busy-dialog, .dialog-pane-busy-indicator, .circular-progress	Busy indicator dialog classes.

```
.dialog-pane > .content-pane {
    -fx-background-radius: 8px;
}
.dialog-pane > .content-pane > .vbox > .button-bar {
    -fx-alignment: center-right;
}
.dialog-pane > .content-pane.error .message-label {
    -fx-font-weight: bold;
}
```

Example CSS for DialogPane.

DialogPane has no custom styleable CSS properties; styling is through ordinary style classes in dialog-pane.css.

11. Localization

Key	English default
button.send	Send
title.default	Dialog
header.title.fallback	Dialog

12. Recipes

12.1 Disable animation

```
dialogs.setAnimateDialogs(false);
dialogs.setFadeInOut(false);
```

12.2 Maximized dialog

```
Dialog<Object> dialog = new Dialog<>(dialogs, DialogPane.Type.INFORMATION);
dialog.setTitle("Details");
dialog.setContent(detailsNode);
dialog.setMaximize(true);
dialog.show();
```

12.3 Required text

```
Dialog<String> dialog = dialogs.showTextInput("Name", "Required", "", false);
dialog.setRequired(true);
```

12.4 Checklist

1. Add DialogPane last so it covers the scene.
2. Use showNode for custom UI and typed helpers for standard cases.
3. Keep long operations outside the JavaFX application thread.
4. Set preferences only for dialogs whose size should persist.

13. See also

- Demo app: `DialogPaneApp`. Run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.DialogPaneApp`.
- Related controls: GlassPane, ResizableTextArea and CircularProgressIndicator usage inside the dialog pane.
- API docs: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/