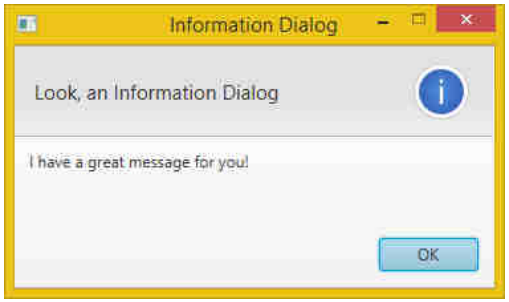
JavaFX Dialogs

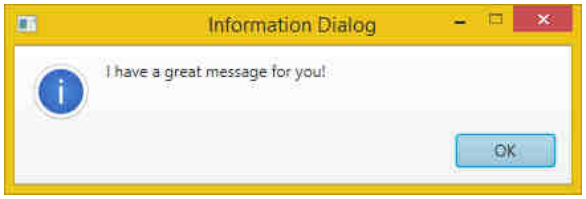
Standard Dialogs

Information Dialog




```
Alert alert = new Alert(AlertType.INFORMATION);
alert.setTitle("Information Dialog");
alert.setHeaderText("Look, an Information Dialog");
alert.setContentText("I have a great message for you!");

alert.showAndWait();
```

Without Header Text

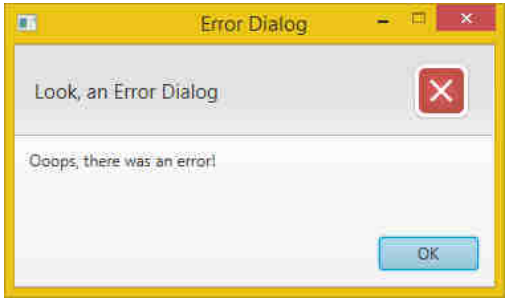



```
Alert alert = new Alert(AlertType.INFORMATION);
alert.setTitle("Information Dialog");
alert.setHeaderText(null);
alert.setContentText("I have a great message for you!");

alert.showAndWait();
```


Warning Dialog

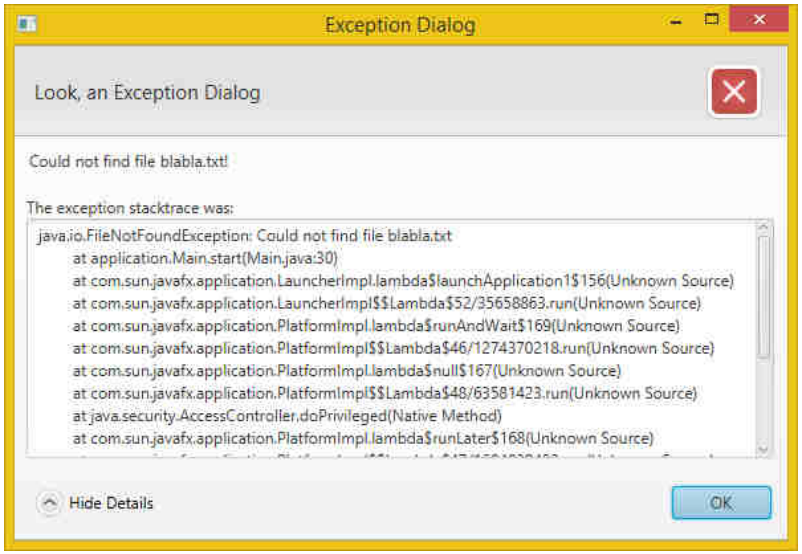

```
Alert alert = new Alert(AlertType.WARNING);
alert.setTitle("Warning Dialog");
alert.setHeaderText("Look, a Warning Dialog");
alert.setContentText("Careful with the next step!");

alert.showAndWait();
```

Error Dialog




```
Alert alert = new Alert(AlertType.ERROR);
alert.setTitle("Error Dialog");
alert.setHeaderText("Look, an Error Dialog");
alert.setContentText("Ooops, there was an error!");

alert.showAndWait();
```

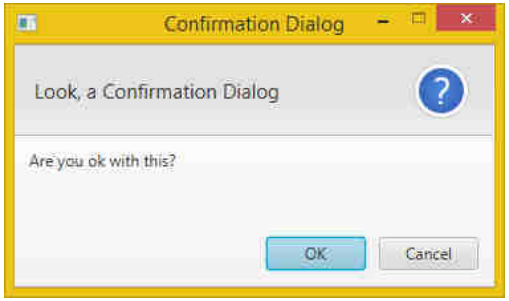
Exception Dialog



There is not a complete Exception Dialog out of the box. But we can easily provide `TextArea` as expandable content.

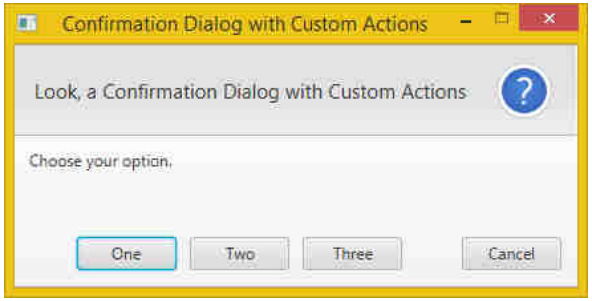

```
Alert alert = new Alert(AlertType.ERROR);
alert.setTitle("Exception Dialog");
alert.setHeaderText("Look, an Exception Dialog");
alert.setContentText("Could not find file blabla.txt!");

Exception ex = new FileNotFoundException("Could not find file blabla.txt");

// Create expandable Exception.
StringWriter sw = new StringWriter();
PrintWriter pw = new PrintWriter(sw);
ex.printStackTrace(pw);
String exceptionText = sw.toString();

Label label = new Label("The exception stacktrace was:");

TextArea textArea = new TextArea(exceptionText);
textArea.setEditable(false);
textArea.setWrapText(true);

textArea.setMaxWidth(Double.MAX_VALUE);
textArea.setMaxHeight(Double.MAX_VALUE);
GridPane.setVgrow(textArea, Priority.ALWAYS);
GridPane.setHgrow(textArea, Priority.ALWAYS);

GridPane expContent = new GridPane();
expContent.setMaxWidth(Double.MAX_VALUE);
expContent.add(label, 0, 0);
expContent.add(textArea, 0, 1);

// Set expandable Exception into the dialog pane.
alert.getDialogPane().setExpandableContent(expContent);

alert.showAndWait();
```

Confirmation Dialog



```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog");
alert.setHeaderText("Look, a Confirmation Dialog");
alert.setContentText("Are you ok with this?");

Optional<ButtonType> result = alert.showAndWait();
if (result.get() == ButtonType.OK){
    // ... user chose OK
} else {
    // ... user chose CANCEL or closed the dialog
}
```

Confirmation Dialog with Custom Actions



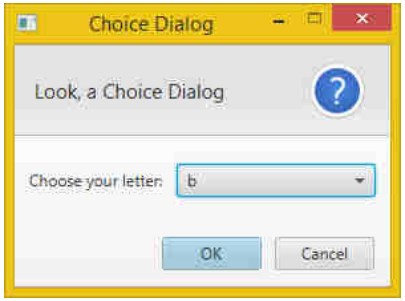

```
Alert alert = new Alert(AlertType.CONFIRMATION);
alert.setTitle("Confirmation Dialog with Custom Actions");
alert.setHeaderText("Look, a Confirmation Dialog with Custom Actions");
alert.setContentText("Choose your option.");

ButtonType buttonTypeOne = new ButtonType("One");
ButtonType buttonTypeTwo = new ButtonType("Two");
ButtonType buttonTypeThree = new ButtonType("Three");
ButtonType buttonTypeCancel = new ButtonType("Cancel", ButtonData.CANCEL_CLOSE);

alert.getButtonTypes().setAll(buttonTypeOne, buttonTypeTwo, buttonTypeThree, buttonTypeCance

Optional<ButtonType> result = alert.showAndWait();
if (result.get() == buttonTypeOne){
    // ... user chose "One"
} else if (result.get() == buttonTypeTwo) {
    // ... user chose "Two"
} else if (result.get() == buttonTypeThree) {
    // ... user chose "Three"
} else {
    // ... user chose CANCEL or closed the dialog
}
```

Text Input Dialog

```
TextInputDialog dialog = new TextInputDialog("walter");
dialog.setTitle("Text Input Dialog");
dialog.setHeaderText("Look, a Text Input Dialog");
dialog.setContentText("Please enter your name:");

// Traditional way to get the response value.
Optional<String> result = dialog.showAndWait();
if (result.isPresent()){
    System.out.println("Your name: " + result.get());
}

// The Java 8 way to get the response value (with lambda expression).
result.ifPresent(name -> System.out.println("Your name: " + name));
```

Note: The `result.isPresent()` will return `false` if the user cancelled the dialog.

Choice Dialog

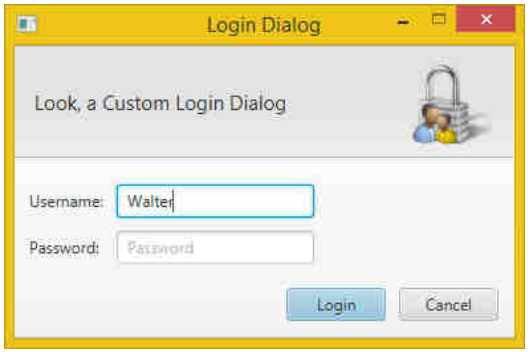



```
List<String> choices = new ArrayList<>();
choices.add("a");
choices.add("b");
choices.add("c");

ChoiceDialog<String> dialog = new ChoiceDialog<>("b", choices);
dialog.setTitle("Choice Dialog");
dialog.setHeaderText("Look, a Choice Dialog");
dialog.setContentText("Choose your letter:");

// Traditional way to get the response value.
Optional<String> result = dialog.showAndWait();
if (result.isPresent()){
    System.out.println("Your choice: " + result.get());
}

// The Java 8 way to get the response value (with lambda expression).
result.ifPresent(letter -> System.out.println("Your choice: " + letter));
```


Note: The `result.isPresent()` will return `false` if the user didn't choose anything or cancelled the dialog.

Custom Login Dialog

Here is an example of how to create a custom dialog with a login form:



```
// Create the custom dialog.
Dialog<Pair<String, String>> dialog = new Dialog<>();
dialog.setTitle("Login Dialog");
dialog.setHeaderText("Look, a Custom Login Dialog");

// Set the icon (must be included in the project).
dialog.setGraphic(new ImageView(this.getClass().getResource("login.png").toString()));

// Set the button types.
ButtonType loginButtonType = new ButtonType("Login", ButtonData.OK_DONE);
dialog.getDialogPane().getButtonTypes().addAll(loginButtonType, ButtonType.CANCEL);

// Create the username and password labels and fields.
GridPane grid = new GridPane();
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(20, 150, 10, 10));

TextField username = new TextField();
username.setPromptText("Username");
PasswordField password = new PasswordField();
password.setPromptText("Password");

grid.add(new Label("Username:"), 0, 0);
grid.add(username, 1, 0);
grid.add(new Label("Password:"), 0, 1);
grid.add(password, 1, 1);

// Enable/Disable login button depending on whether a username was entered.
Node loginButton = dialog.getDialogPane().lookupButton(loginButtonType);
loginButton.setDisable(true);

// Do some validation (using the Java 8 lambda syntax).
username.textProperty().addListener((observable, oldValue, newValue) -> {
    loginButton.setDisable(newValue.trim().isEmpty());
});

dialog.getDialogPane().setContent(grid);

// Request focus on the username field by default.
Platform.runLater(() -> username.requestFocus());

// Convert the result to a username-password-pair when the login button is clicked.
dialog.setResultConverter(dialogButton -> {
    if (dialogButton == loginButtonType) {
        return new Pair<>(username.getText(), password.getText());
    }
    return null;
});
```



```
Optional<Pair<String, String>> result = dialog.showAndWait();

result.ifPresent(usernamePassword -> {
    System.out.println("Username=" + usernamePassword.getKey() + ", Password=" + usernamePas
});
```

Styling the Dialogs

Custom Icon

In the current version it's a bit cumbersome to get to the Dialog's `Stage` to be able to set its icon. Here is how:

```
// Get the Stage.
Stage stage = (Stage) dialog.getDialogPane().getScene().getWindow();

// Add a custom icon.
stage.getIcons().add(new Image(this.getClass().getResource("login.png").toString()));
```

According to this bug report (https://javafx-jira.kenai.com/browse/RT-38895) the final version of the JavaFX 8u40 Dialogs should use the same icon as the application that it is running from. In that case you would need to set its owner and the Dialog would get the owner's icon:

```
dialog.initOwner(otherStage);
```

Minimal Decorations

Another option is to remove the icon and use only minimal window decorations.


```
dialog.initStyle(StageStyle.UTILITY);
```

Other Options

Setting the Owner

You can specify the owner `Window` for a dialog. If no owner or null is specified for the owner, it is a top-level, unowned dialog.

```
dialog.initOwner(parentWindow);
```

Setting the Modality

You can specify the modality for a dialog. The modality must be one of `Modality.NONE`, `Modality.WINDOW_MODAL`, or `Modality.APPLICATION_MODAL`.

```
dialog.initModality(Modality.NONE);
```

API Documentation

For more information on the Dialogs have a look at the JavaFX API docs:

- Alert (http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Alert.html)
- Dialog (http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Dialog.html)
- TextInputDialog (http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextInputDialog.html)
- ChoiceDialog (http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ChoiceDialog.html)
- DialogPane (http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/DialogPane.html)
