ĐỐI TƯỢNG KHUNG CHỨA TRONG JAVAFX

Về mặt khái niệm khung chứa là các thành phần mà có thể chứa các thành phần khác. Về cách tổ chức giao diện, Java AWT và Swing tách biệt khung chứa và bộ quản lý trình bày (layout). Một bộ quản lý trình bày trong Java AWT và Swing chỉ có ý nghĩa logic, các thành phần con được thêm trực tiếp vào một số bộ khung chứa như frame, dialog, panel.

JavaFX thì khác, nó tổ chức giao diện thành các nút (Node), mỗi nút ứng với một thành phần giao diện, nút này chứa trong nó nút kia, cứ như vậy cho đến thành phần con nhỏ nhất. Các bộ quản lý trình bày trong JavaFX cũng là các nút, vừa có tác dụng khung chứa, vừa có tác dụng quản lý trình bày các thành phần giao diện con.

Trong phần này, chúng tôi tập trung giới thiệu một số đối tượng khung chứa có dạng một cửa sổ, hộp thoại. Các khung chứa có chức năng của một bộ quản lý trình bày được chúng tôi trình bày ở phần sau.

▪ Stage

Lớp javafx.scene.stage.Stage đã được giới thiệu trong phần mở đầu về JavaFX. Trong phần này ta sẽ tìm hiểu cách mở một cửa sổ Stage mới như thế nào.

Từ một cửa sổ Stage này, ta có thể mở một cửa sổ Stage mới bằng lệnh:

stage.initModality(Modelity modality)

Trong đó "modelity" là tham số cho phép ta mở một cửa sổ Stage mới theo 3 mẫu:

+ Modelity.NONE: Khi ta mở một cửa sổ mới với tùy chọn này, cửa sổ mới sẽ độc lập với cửa sổ cha. Ta có thể tương tác với cửa sổ cha, hoặc đóng cửa sổ cha mà không ảnh hưởng tới cửa sổ mới. Đây là chế độ mặc định cho một Stage mới được tạo ra từ một sự kiện nào đó.

+ Modelity.WINDOW_MODAL: Khi ta mở một cửa sổ mới với tùy chọn này, cửa sổ mới sẽ khóa cửa sổ cha. Ta không thể tương tác với cửa sổ cha, cho tới khi cửa sổ này bị đóng lại. Trong trường hợp này, ta cần chú ý xác định cửa sổ cha cho cửa sổ mới bằng phương thức: stage.initOwner(Stage parentStage)

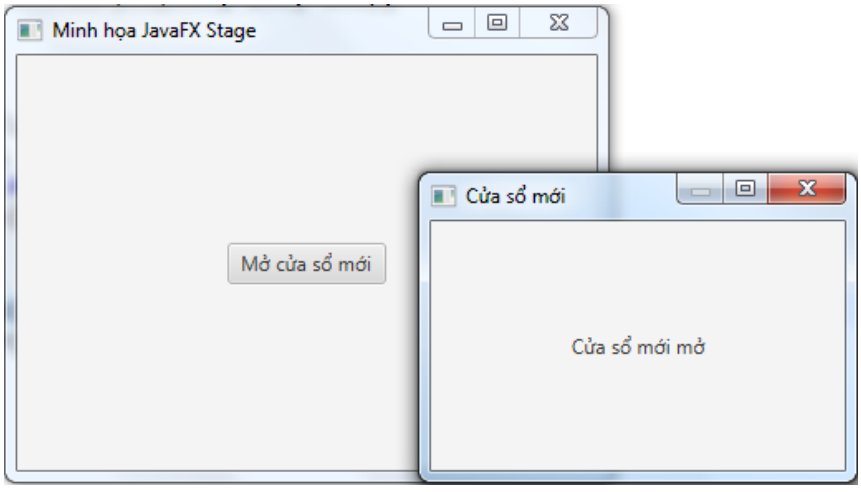
+ Modelity.APPLICATION_MODAL: Khi ta mở một cửa sổ mới với tùy chọn này, nó sẽ khóa mọi cửa sổ khác của ứng dụng. Ta không thể tương tác với bất kỳ cửa sổ nào khác cho tới khi cửa sổ này bị đóng lại

Ví dụ:

```
@Override
    public void start(final Stage primaryStage) {

        Button button = new Button("Mở cửa sổ mới");

        button.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                StackPane secondLayout = new StackPane();
                secondLayout.getChildren().add(new Label("Cửa sổ mới mở"));
```

```
                Scene secondScene = new Scene(secondLayout, 250, 150);

                // Một cửa sổ mới (Stage)
                Stage newStage = new Stage();
                newStage.setTitle("Cửa sổ mới");
                newStage.setScene(secondScene);
                newStage.initModality(Modality.WINDOW_MODAL);
                newStage.initOwner(primaryStage);

                // Đặt vị trí cho cửa sổ thứ 2.
                newStage.setX(primaryStage.getX() + 250);
                newStage.setY(primaryStage.getY() + 100);

                newStage.show();
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(button);

        Scene scene = new Scene(root, 350, 250);

        primaryStage.setTitle("Minh họa JavaFX Stage");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

Kết quả chạy chương trình:



▪ Alert Dialog

Lớp **Alert** thừa kế từ lớp **Dialog** (Hộp thoại) cung cấp hỗ trợ cho một số loại **hộp thoại** để ta có thể dễ dàng hiển thị cho người dùng nhắc nhở về một phản hồi.

JavaFX cung cấp 4 loại hộp thoại nhắc nhở bao gồm: hộp thoại thông tin (information alert), hộp thoại xác nhận (confirm alert), hộp thoại cảnh báo (warning alert) và hộp thoại báo

lỗi (error alert). Để tạo ra các đối tượng hộp thoại này, ta dùng một constructor, cùng tham số chỉ loại hộp thoại:

```
Alert alert = new Alert(AlertType.INFORMATION);
Alert alert = new Alert(AlertType.CONFIRMMATION);
Alert alert = new Alert(AlertType.WARNING);
Alert alert = new Alert(AlertType.ERROR);
```

Nội dung chứa trong một hộp thoại Alert bao gồm 3 phần:

+ Phần tiêu đề nhắc nhở (header): phần header là phần cấu hình tùy chọn. Phương thức sử dụng với header:

- *alert.setHeaderText(String headerText)*: đặt nội dung cho header, nếu không gọi phương thức này, header được hiển thị theo chế độ mặc định
- *alert.setHeaderText(null)*: không hiển thị khối header

+ Phần nội dung (content): chứa nội dung cần nhắc nhở. Phương thức sử dụng với content:

- *alert.setContentText(String contentText)*

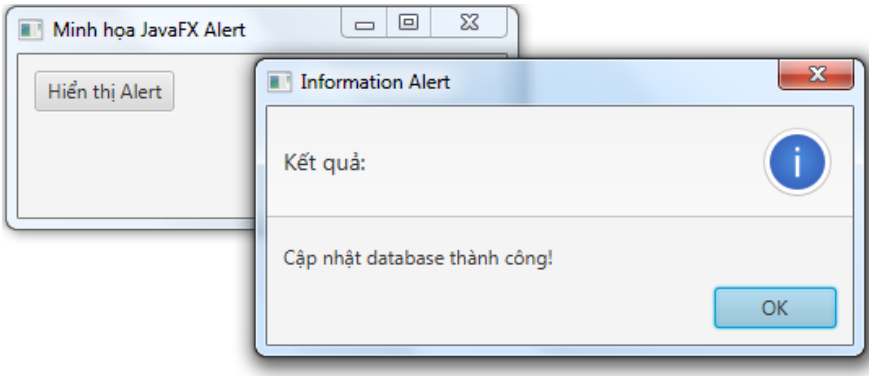
+ Phần chân trang (footer): thường chứa đựng các nút nhấn có sẵn, các nút nhấn này có thể thay đổi được

Ví dụ:

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class AlertDemo extends Application {

    private void showAlert() {
        Alert alert = new Alert(AlertType.INFORMATION);
        alert.setTitle("Information Alert");
        alert.setHeaderText("Kết quả:");
        alert.setContentText("Cập nhật database thành công!");

        alert.showAndWait();
    }

    @Override
    public void start(Stage stage) {
```

```
        VBox root = new VBox();
        root.setPadding(new Insets(10));
        Button btn = new Button("Hiển thị Alert");

        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                showAlert();
            }
        });

        root.getChildren().add(btn);

        Scene scene = new Scene(root, 300, 100);
        stage.setTitle("Minh họa JavaFX Alert");
        stage.setScene(scene);

        stage.show();

    }

    public static void main(String args[]) {
        launch(args);
    }

}
```

Kết quả chạy chương trình:



▪ ChoiceDialog

ChoiceDialog là một lớp con của lớp **Dialog**, nó hiển thị một danh sách các lựa chọn cho người dùng, và người dùng có thể chọn nhiều nhất một lựa chọn.

Về mặt cấu trúc, ChoiceDialog bao gồm các thành phần tương tự như một hộp thoại Alert, nó có thêm một ChoiceBox cho người dùng lựa chọn.

ChoiceDialog có thể được tạo bởi một số cách:

ChoiceDialog()

ChoiceDialog(T defaultChoice, Collection<T> choices)

ChoiceDialog(T defaultChoice, T... choices)

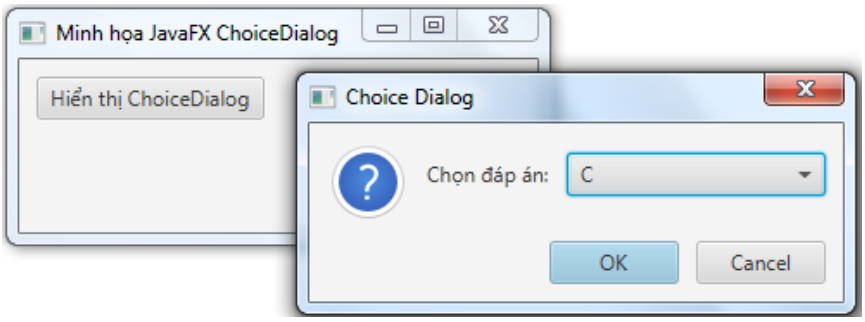
Trong đó "defaultChoice" là một lựa chọn được chỉ định để mặc định khi hiển thị, nó là một thành phần trong danh sách các lựa chọn.

Ví dụ:

```
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ChoiceDialog;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class ChoiceDialogDemo extends Application{

	private Label choiceLabel;
	private void showChoiceDialog() {
		List<String> choices = new ArrayList<>();
		choices.add("A");
		choices.add("B");
		choices.add("C");
		choices.add("D");

		ChoiceDialog<String> dialog =
			new ChoiceDialog<String>("B", choices);
		dialog.setTitle("Choice Dialog");
		dialog.setHeaderText(null);
		dialog.setContentText("Chọn đáp án:");

		Optional<String> result = dialog.showAndWait();
		if (result.isPresent()){
			choiceLabel.setText("Lựa chọn: " + result.get());
		}
	}

	@Override
	public void start(Stage stage) {

	    choiceLabel = new Label();
		VBox root = new VBox();
```

```
        root.setPadding(new Insets(10));
        Button btn = new Button("Hiển thị ChoiceDialog");

        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                showChoiceDialog();
            }
        });

        root.getChildren().addAll(btn, choiceLabel);

        Scene scene = new Scene(root, 300, 100);
        stage.setTitle("Minh họa JavaFX ChoiceDialog");
        stage.setScene(scene);

        stage.show();

    }

    public static void main(String args[]) {
        launch(args);
    }
}
```

Kết quả chạy chương trình:



▪ TextInputDialog

TextInputDialog là một lớp con của lớp **Dialog**, nó được sử dụng để hiển thị và chờ đợi người dùng nhập vào một nội dung văn bản.

Về mặt cấu trúc, ChoiceDialog bao gồm các thành phần tương tự như một hộp thoại Alert, nó có thêm một TextField cho người dùng nhập vào nội dung văn bản.

Có 2 cách tạo đối tượng TextInputDialog:

TextInputDialog()

TextInputDialog(String defaultValue)

Nếu một tham số chuỗi ký tự được sử dụng trong constructor, chuỗi ký tự sẽ được sử dụng để khởi tạo giá trị cho TextField bên trong TextInputDialog.

Ví dụ:

```
TextInputDialog dialog = new TextInputDialog();
dialog.setTitle("Text Input Dialog");
dialog.setHeaderText(null);
dialog.setContentText("Nhập mã sinh viên:");

Optional<String> result = dialog.showAndWait();
if (result.isPresent()){
    System.out.println("Msv vừa nhập: " + result.get());
}
```

Kết quả chạy chương trình (sau khi hoàn thiện):