CSCI 2120:
Software Design & Development II

UNIT4: UI management

GUI framework

JavaFX Overview

Overview

1. Introduction
2. Stage
3. Scene
   a. Scene Graph
   b. Nodes
4. Controls
5. Layouts
   a. Nested Layouts
6. Charts
7. 2D Graphics
8. 3D Graphics
9. Audio
10. Video
11. WebView

Introduction

- To fully benefit from JavaFX it is useful to understand how JavaFX is designed, and to have a good overview of what features JavaFX contains. The purpose of this text is to give you that JavaFX overview. This text will first look at the general JavaFX design, then look at the various features in JavaFX.

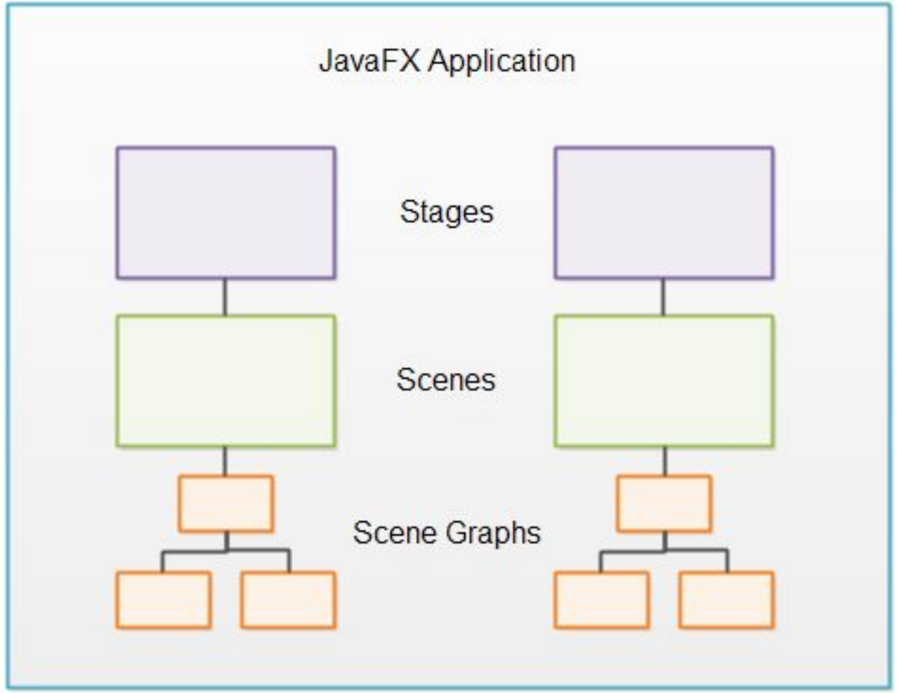
- If you are familiar with Flash / Flex, you will see that JavaFX is somewhat inspired by Flash / Flex. Some of the same ideas are found in JavaFX.

- In general, a JavaFX application contains one or more stages which corresponds to windows. Each stage has a scene attached to it. Each scene can have an object graph of controls, layouts etc. attached to it, called the scene graph.

- These concepts are all explained in more detail later.

Introduction

Here is an illustration of the general structure of a JavaFX application:



Stage

The *stage* is the outer frame for a JavaFX application. The stage typically corresponds to a window. In the early days where JavaFX could run in the browser, the stage could also refer to the area inside the web page that JavaFX had available to draw itself.

Since the deprecation of the Java browser plugin JavaFX is mostly used for desktop applications. Here, JavaFX replaces Swing as the recommended desktop GUI framework. And I must say, that JavaFX looks a whole lot more consistent and feature rich than Swing.

When used in a desktop environment, a JavaFX application can have multiple windows open. Each window has its own stage.

Each stage is represented by a Stage object inside a JavaFX application. A JavaFX application has a primary Stage object which is created for you by the JavaFX runtime. A JavaFX application can create additional Stage objects if it needs additional windows open. For instance, for dialogs, wizards etc.

Scene

To display anything on a stage in a JavaFX application, you need a *scene*. A stage can only show one scene at a time, but it is possible to exchange the scene at runtime. Just like a stage in a theater can be rearranged to show multiple scenes during a play, a stage object in JavaFX can show multiple scenes (one at a time) during the life time of a JavaFX application.

You might wonder why a JavaFX application would ever have more than one scene per stage. Imagine a computer game. A game might have multiple "screens" to show to the user. For instance, an initial menu screen, the main game screen (where the game is played), a game over screen and a high score screen. Each of these screens can be represented by a different scene. When the game needs to change from one screen to the next, it simply attaches the corresponding scene to the Stage object of the JavaFX application.

A scene is represented by a Scene object inside a JavaFX application. A JavaFX application must create all Scene objects it needs.

Scene → Scene Graph

All visual components (controls, layouts etc.) must be attached to a scene to be displayed, and that scene must be attached to a stage for the whole scene to be visible. The total object graph of all the controls, layouts etc. attached to a scene is called the *scene graph*.

Scene → Nodes

All components attached to the scene graph are called *nodes*. All nodes are subclasses of a JavaFX class called javafx.scene.Node .

There are two types of nodes: Branch nodes and leaf nodes. A branch node is a node that can contain other nodes (child nodes). Branch nodes are also referred to as parent nodes because they can contain child nodes. A leaf node is a node which cannot contain other nodes.

Controls

JavaFX controls are JavaFX components which provide some kind of control functionality inside a JavaFX application. For instance, a button, radio button, table, tree etc.

For a control to be visible it must be attached to the scene graph of some Scene object.

Controls are usually nested inside some JavaFX layout component that manages the layout of controls relative to each other.

JavaFX contains the following controls:

Accordion Button CheckBox ChoiceBox ColorPicker ComboBox	DatePicker Label ListView Menu MenuBar PasswordField	ProgressBar RadioButton Slider Spinner SplitMenuButton	SplitPane TableView TabPane TextArea TextField	TitledPane ToggleButton ToolBar TreeTableView TreeView

Layouts

JavaFX layouts are components which contains other components inside them. The layout component manages the layout of the components nested inside it. JavaFX layout components are also sometimes called *parent components* because they contain child components, and because layout components are subclasses of the JavaFX class javafx.scene.Parent.

A layout component must be attached to the scene graph of some Scene object to be visible.

JavaFX contains the following layout components:

Group Region Pane HBox	VBox FlowPane BorderPane StackPane	TilePane GridPane AnchorPane TextFlow

Layouts → Nested Layouts

It is possible to nest layout components inside another layout components. This can be useful to achieve a specific layout. For instance, to get horizontal rows of components which are not laid out in a grid, but differently for each row, you can nest multiple HBox layout components inside a VBox component.

Charts

JavaFX comes with a set of built-in ready-to-use chart components, so you don't have to code charts from scratch every time you need a basic chart.

JavaFX contains the following chart components:

- AreaChart
- BarChart
- BubbleChart
- LineChart
- PieChart
- ScatterChart
- StackedAreaChart
- StackedBarChart

2D Graphics

JavaFX contains features that makes it easy to draw 2D graphics on the screen.

3D Graphics

JavaFX contains features that makes it easy to draw 3D graphics on the screen.

Audio

JavaFX contains features that makes it easy to play audio in JavaFX applications. This is typically useful in games or educational applications.

Video

JavaFX contains features that makes it easy to play video in JavaFX applications. This is typically useful in streaming applications, games or educational applications.

WebView

JavaFX contains a WebView component which is capable of showing web pages (HTML5, CSS etc.). The JavaFX WebView component is based on WebKit - the web page rendering engine also used in Chrome and Safari.

The WebView component makes it possible to mix a desktop application with a web application. There are times where that is useful. For instance, if you already have a decent web application, but need some features which can only be provided sensibly with a desktop application - like disk access, communication with other network protocols than HTTP (e.g UDP, IAP etc.) .

END