VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

JavaFX Introduction

Objectives

This session will give the knowledge about

- Introduction to JavaFX

What is JavaFX?

JavaFX is a Java library that is used to develop Desktop applications as well as Rich Internet Applications (RIA).

The applications built in JavaFX, can run on multiple platforms including Web, Mobile and Desktops.

JavaFX is intended to replace swing in Java applications as a GUI framework. However, It provides more functionalities than swing.

Like Swing, JavaFX also provides its own components and It is lightweight and hardware accelerated. It is independent of operating systems.

What is RIA?

Rich Internet Applications are those web applications which provide similar features and experience as that of desktop applications. They offer a better visual experience.

RIA's don't require to have any additional software to run. As an alternative, you should install software such as ActiveX, Java, Flash. The following three main technologies using which we can develop an RIA:

- Adobe Flash
- Microsoft Silverlight
- JavaFX

History of JavaFX

- JavaFX was developed by Chris Oliver. Initially the project was named as Form Follows Functions (F3). It is intended to provide the richer functionalities for the GUI application development.
- Later, Sun Micro-systems acquired F3 project as JavaFX in June, 2005. Sun Micro-systems announces it officially in 2007 at W3 Conference.
- In October 2008, JavaFX 1.0 was released. In 2009, ORACLE corporation acquires Sun Micro-Systems and released JavaFX 1.2. the latest version of JavaFX is JavaFX 15 (https://openjfx.io/).
- From jdk11, JavaFX has been removed and made as separate JDK.

Features of JavaFX

- Java Library — It is a Java library which consists of many classes and interfaces that are written in Java.
- FXML — FXML is the XML based Declarative mark up language. The coding can be done in FXML to provide the more enhanced GUI to the user.
- Scene Builder — Scene Builder generates FXML mark-up which can be ported to an IDE.

Features of JavaFX

- **Web view** — Web pages can be embedded with JavaFX applications. Web View uses WebKitHTML technology to embed web pages.
- **Built in UI controls** — JavaFX contains Built-in components which are not dependent on operating system. The UI component are just enough to develop a full featured application.
- **CSS like styling** — JavaFX code can be embedded with the CSS to improve the style of the application.

Features of JavaFX

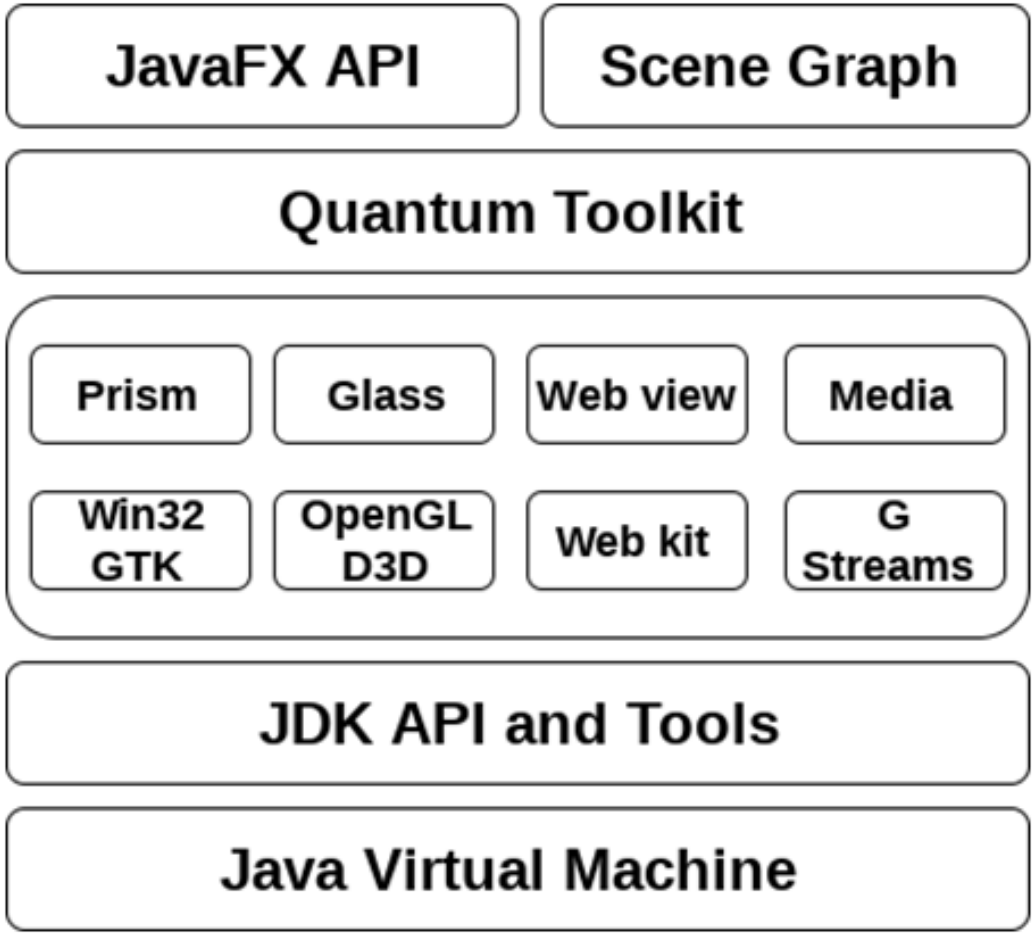
- **Swing interoperability** — The JavaFX applications can be embedded with swing code using the Swing Node class. We can update the existing swing application with the powerful features of JavaFX.
- **Canvas API** — Canvas API provides the methods for drawing directly in an area of a JavaFX scene.
- **Rich Set of APIs** — JavaFX provides a rich set of API's to develop GUI applications.

Features of JavaFX

- **Integrated Graphics Library** — An integrated set of classes are provided to deal with 2D and 3D graphics.
- **Graphics Pipeline** — JavaFX graphics are based on Graphics rendered pipeline(prism). It offers smooth graphics which are hardware accelerated.
- **High Performance Media Engine** — The media pipeline supports the playback of web multimedia on a low latency. It is based on a Gstreamer Multimedia framework.

JavaFX Architecture

- This image shows the complete architecture of JavaFX platform.
- There are various built-in components which are interconnected with each other.
- However, JavaFX contains a rich set of APIs to develop rich internet applications which run consistently across many platforms.



JavaFX public API

The top layer of JavaFX architecture contains a JavaFX public API which provides all the necessary classes that are responsible for executing a full featured JavaFX application.

The list of all the packages of this API are as follows.

- javafx.animation — Provides the set of classes that are responsible for transitions based animations
- javafx.application — Provides application life-cycle methods
- javafx.collections — Provides classes that can handle collections and related utilities

JavaFX public API

- javafx.concurrent — Provides classes that are responsible for multitasking
- javafx.embed.swing — Provides the set of classes that can be used inside the swing code
- javafx.embed.swt — Provides the set of classes that can be used inside the swt code
- javafx.event — Provides the classes that deal with events and their handling

JavaFX public API

• javafx.fxml	Contains the set of classes that are responsible of loading hierarchy from mark-up
• javafx.geometry	Provides the 2D classes that contains the methods to operate 2D geometry on the object.
• javafx.scene	Provides the classes to deal with scene graph API
• javafx.scene.canvas	Provides the set of classes that deal with canvas.
• javafx.scene.control	Contains the classes for all JavaFX components.
• javafx.scene.effect	Contains the set of classes that apply the graphic effects to scene graph nodes

JavaFX public API

- javafx.scene.image — Provides the set of classes for loading and displaying images
- javafx.scene.input — Provides the set of classes for the mouse and keyboard events
- javafx.scene.layout — Provides the set of classes to support user interface layout
- javafx.scene.media — Provides the set of classes to integrate audio and video into JavaFX application

JavaFX public API

- javafx.scene.paint — Provides the set of classes for colours and gradients to fill shapes and backgrounds when rendering scene graph.
- javafx.scene.shape — Provides the set of 2D classes that performs the operations on objects related to 2D geometry.
- javafx.scene.text — Provides the set of classes for fonts and rendering text nodes.
- javafx.scene.transform — Provides the set of classes that are used to perform rotating, scaling, shearing operations on objects.

JavaFX public API

- javafx.scene.web — Provides means for loading and displaying web content.
- javafx.stage — Provides the top level container classes for JavaFX content.
- javafx.util — Provides utilities classes
- javafx.util.converter — This package is for standard string converters for JavaFX

Scene Graph

- It is the starting point of constructing a JavaFX application.
- It is a hierarchical tree of nodes that represent all the visual elements of user interface. It also have the capability of handling event. In general, scene graph can be defined as a collection of nodes.
- Each node has its separate id, style and volume. Every node of a scene graph can only have single parent and zero or more children.
- All the implementation on a scene graph are actually applied to its node. Their are various classes present in javafx.scene package that are used for creating, modifying and applying some transformations on the node.

Graphics Engine

- The JavaFX graphics engine provides the graphics support to the scene graph. It basically supports 2D as well as 3D graphics both.
- It provides the software rendering when the graphics hardware present on the system is not able to support hardware accelerated rendering.
- The two graphics accelerated pipelines in the JavaFX are:
    - Prism
    - Quantum Tool kit

Graphics Engine

Prism

- prism can be seen as High Performance hardware-accelerated graphics pipeline. It has the capability to render both 2D and 3D graphics. Prism implements different ways to render graphics on different platforms.

Quantum Tool kit

- Quantum Tool Kit is used to bind prism and glass windowing tool kit together and makes them available for the above layers in stack.

Glass Windowing tool kit

- It is present on the lowest level of JavaFX graphics stack.
- It basically can be seen as a platform dependent layer which works as an interface between JavaFX platform and native operating system.
- It is responsible for providing the operating system services such as managing the windows, timers, event queues and surfaces.

Web View

- We can also embed the HTML content to a JavaFX scene graph. For this purpose, JavaFX uses a component called web view.
- Web view uses web kit which is an internal open source browser and can render HTM5, DOM, CSS, SVG and JavaScript.
- Using web view, we can render the HTML content from JavaFX application , and also apply some CSS styles to the user interface.

Media Engine

- By using Media engine, the JavaFX application can support the playback of audio and video media files.
- JavaFX media engine depends upon an open source engine called as G Streamer.
- The JavaFX media engine provides support for audio for the following file formats.

Audio	• MP3 • WAV • AIFF	Video	• FLV

Media Engine

- The package javafx.scene.media contains the classes and interfaces to provide media functionality in JavaFX.
- It is provided in the form of three components, which are
    - Media Object − This represents a media file
    - Media Player − To play media content.
    - Media View − To display media.

JavaFX Application Structure

JavaFX application is divided hierarchically into three main components known as Stage, Scene and nodes.

We need to import javafx.application.Application class in every JavaFX application. This provides the following life cycle methods for JavaFX application.

- public void init()
- public abstract void start(Stage primaryStage)
- public void stop()

JavaFX Application Structure

- start() − The entry point method where the JavaFX graphics code is to be written.
- stop() − An empty method which can be overridden, here you can write the logic to stop the application.
- init() − An empty method which can be overridden, but you cannot create stage or scene in this method.

In addition to these, it provides a static method named launch() to launch JavaFX application.

JavaFX application Steps:

In order to create a basic JavaFX application, we need to:

1. Import javafx.application.Application into our code.
2. Inherit Application into our class.
3. Override start() method of Application class.
4. Call launch() method from main method.

MyFirst JavaFX application:

Stage

- Stage in a JavaFX application is similar to the Frame in a Swing Application.
- It acts like a container for all the JavaFX objects. Primary Stage is created internally by the platform. The object of primary stage is passed to start method.
- We need to call show method on the primary stage object in order to show our primary stage. Initially, the primary Stage looks like following.

MyFirst JavaFX application:

Scene

- Scene actually holds all the physical contents (nodes) of a JavaFX application. Javafx.scene.Scene class provides all the methods to deal with a scene object.
- Creating scene is necessary in order to visualize the contents on the stage.
- The Scene can be created by creating the Scene class object and passing the layout object into the Scene class constructor.

MyFirst JavaFX application:

Layouts

- Layouts are the top level container classes that define the UI styles for scene graph objects. Layout is the parent node to all the other nodes.
- All these classes belong to javafx.scene.layout package. javafx.scene.layout.Pane class is the base class for all the built-in layout classes in JavaFX.
- We have several built-in layout panes in JavaFX that are HBox, VBox, StackPane, FlowBox, AnchorPane, etc.

MyFirst JavaFX application:

Controls

- The UI elements are the one which are actually shown to the user for interaction or information exchange.
- Layout defines the organization of the UI elements on the screen. Behaviour is the reaction of the UI element when some event is occurred on it.
- The package javafx.scene.control provides all the necessary classes for the UI components like Button, Label, etc. Every class represents a specific UI control and defines some methods for their styling.

MyFirst JavaFX application:

Events

- In JavaFX, events are basically used to notify the application about the actions taken by the user. JavaFX provides the mechanism to capture the events, route the event to its target and letting the application to handle it.
- JavaFX provides the class javafx.event.Event which contains all the subclasses representing the types of Events that can be generated.
- There are various events in JavaFX i.e. MouseEvent, KeyEvent, ScrollEvent, DragEvent, etc. We can also define our own event by inheriting the class javafx.event.Event.

MyFirst JavaFX Example:

Step 1: Extend javafx.application.Application and override start()

```
import javafx.application.Application;
import javafx.stage.Stage;
public class FirstFXDemo extends Application{
    @Override
    public void start(Stage primaryStage) throws Exception {
        // TODO Auto-generated method stub

    }
}
```

MyFirst JavaFX Example:

Step 2: Create a Button

```
import javafx.scene.control.Button;
public class FirstFXDemo extends Application{
    @Override
    public void start(Stage primaryStage) throws Exception {
        Button b1=new Button("click me");
    }
}
```

MyFirst JavaFX Example:

Step 3: Create a layout and add button to it

```
import javafx.scene.layout.StackPane;

public void start(Stage primaryStage) throws Exception {
        Button b1=new Button("click me");
        StackPane root=new StackPane();
        root.getChildren().add(btn1);
}
```

MyFirst JavaFX Example:

Step 4: Create a Scene

```
import javafx.scene.Scene;

public void start(Stage primaryStage) throws Exception {
        Button b1=new Button("click me");
        StackPane root=new StackPane();
        root.getChildren().add(btn1);
        Scene scene=new Scene(root);
}
```

MyFirst JavaFX Example:

Step 5: Prepare the Stage

```
import javafx.scene.Scene;

public void start(Stage primaryStage) throws Exception {
        .....
        primaryStage.setScene(scene);
        primaryStage.setTitle("First JavaFX Application");
        rimaryStage.show();

}
```

MyFirst JavaFX Example:

Step 6: Create the main method

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

Set ->Run Configurations->Arguments->VM Arguments

--module-path "C:\Program Files\javafx-sdk-18.0.1\lib" --add-modules=javafx.controls,javafx.fxml

e(fx)clipse

Summary

We have discussed about

- JavaFX Introduction