JavaFX

# Introducción

Las aplicaciones de Internet enriquecidas (RIA por sus siglas en inglés) son aquellas aplicaciones web que proporcionan características y experiencias similares a las de las aplicaciones de escritorio. Ofrecen a los usuarios una mejor experiencia visual en comparación con las aplicaciones web normales. Estas aplicaciones se entregan como complementos del navegador o como máquina virtual y se utilizan para transformar las aplicaciones estáticas tradicionales en aplicaciones más mejoradas, fluidas, animadas y atractivas.

A diferencia de las aplicaciones de escritorio tradicionales, las RIA no requieren tener ningún software adicional para funcionar. Como alternativa, se debe instalar software como ActiveX, Java, Flash, dependiendo de la aplicación.

En una RIA, la presentación gráfica se maneja en el lado del cliente, ya que tiene un plugin que proporciona soporte para gráficos ricos. En pocas palabras, la manipulación de datos en una RIA se lleva a cabo en el lado del servidor, mientras que la manipulación de objetos relacionados se lleva a cabo en el lado del cliente.

## ¿Qué es JavaFX?

JavaFX es una biblioteca de Java que se utiliza para crear RIA. Las aplicaciones escritas con esta biblioteca pueden ejecutarse de forma consistente en múltiples plataformas. Las aplicaciones desarrolladas con JavaFX pueden ejecutarse en varios dispositivos, como ordenadores de sobremesa, teléfonos móviles, televisores, tabletas, etc.

Para desarrollar aplicaciones del lado del cliente con características enriquecidas, los programadores solían depender de varias bibliotecas para añadir características como medios, controles de interfaz de usuario, web, 2D y 3D, etc. JavaFX incluye todas estas características en una única biblioteca. Además, los desarrolladores también pueden acceder a las características existentes de una biblioteca Java como Swing.

JavaFX ofrece un rico conjunto de APIs gráficas y multimedia y aprovecha la moderna Unidad de Procesamiento Gráfico mediante gráficos acelerados por hardware. JavaFX también proporciona interfaces con las que los desarrolladores pueden combinar la animación de gráficos y el control de la interfaz de usuario.

Se puede utilizar JavaFX con tecnologías basadas en JVM como Java, Groovy y JRuby. Si los desarrolladores optan por JavaFX, no hay necesidad de aprender tecnologías adicionales, ya que el conocimiento previo de cualquiera de las tecnologías mencionadas será suficiente para desarrollar RIA utilizando JavaFX.

## Características de JavaFX

A continuación, se indican algunas de las características importantes de JavaFX.

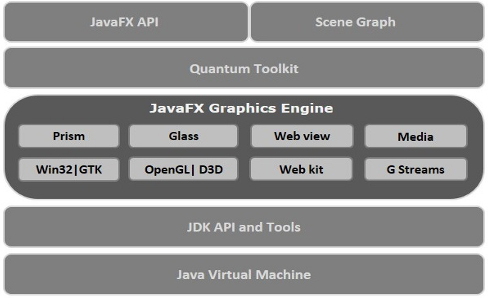
* Escrito en Java - La biblioteca de JavaFX está escrita en Java y está disponible para los lenguajes que se pueden ejecutar en una JVM, que incluyen - Java, Groovy y JRuby. Estas aplicaciones JavaFX también son independientes de la plataforma.
* FXML - JavaFX cuenta con un lenguaje conocido como FXML, que es un lenguaje de marcado declarativo similar al HTML. El único propósito de este lenguaje es definir una interfaz de usuario.
* Scene Builder - JavaFX proporciona una aplicación llamada Scene Builder. Al integrar esta aplicación en IDE's como Eclipse y NetBeans, los usuarios pueden acceder a una interfaz de diseño de arrastrar y soltar, que se utiliza para desarrollar aplicaciones FXML (al igual que las aplicaciones Swing Drag & Drop y DreamWeaver).
* Interoperabilidad con Swing - En una aplicación JavaFX, se puede incrustar contenido Swing utilizando la clase Swing Node. Del mismo modo, puede actualizar las aplicaciones Swing existentes con las características de JavaFX, como el contenido web incrustado y los medios gráficos enriquecidos.
* Controles de interfaz de usuario incorporados: la biblioteca JavaFX ofrece controles de interfaz de usuario con los que se puede desarrollar una aplicación completa.
* Estilos similares a los de CSS: JavaFX proporciona estilos similares a los de CSS. Al usar esto, puede mejorar el diseño de su aplicación con un simple conocimiento de CSS.
* API de lienzo e impresión - JavaFX proporciona lienzo, un estilo de modo inmediato de la API de renderizado. Dentro del paquete javafx.scene.canvas contiene un conjunto de clases para canvas, con las que podemos dibujar directamente dentro de un área de la escena de JavaFX. JavaFX también proporciona clases para la impresión en el paquete javafx.print.
* Rico conjunto de API's - La biblioteca JavaFX proporciona un rico conjunto de API's para desarrollar aplicaciones GUI, gráficos 2D y 3D, etc. Este conjunto de API también incluye las capacidades de la plataforma Java. Por lo tanto, utilizando esta API, se puede acceder a las características de los lenguajes Java como los genéricos, las anotaciones, el multithreading y las expresiones Lambda.
* Biblioteca de gráficos integrada - JavaFX proporciona clases para gráficos 2d y 3d.
* Canalización de gráficos - JavaFX admite gráficos basados en la canalización de gráficos acelerada por hardware conocida como Prism. Cuando se utiliza con una tarjeta gráfica o una GPU compatible, ofrece gráficos fluidos. En caso de que el sistema no soporte la tarjeta gráfica, el prisma se basa en la pila de renderizado por software.

## Arquitectura

### Esquema General

JavaFX proporciona una completa API con un rico conjunto de clases e interfaces para construir aplicaciones GUI con gráficos ricos. Los paquetes importantes de esta API son:

* javafx.animation - Contiene clases para añadir animaciones basadas en transiciones, como relleno, desvanecimiento, rotación, escala y traslación, a los nodos JavaFX.
* javafx.application - Contiene un conjunto de clases responsables del ciclo de vida de la aplicación JavaFX.
* javafx.css - Contiene clases para añadir estilos tipo CSS a las aplicaciones JavaFX GUI.
* javafx.event - Contiene clases e interfaces para entregar y manejar eventos JavaFX.
* javafx.geometry - Contiene clases para definir objetos 2D y realizar operaciones con ellos.
* javafx.stage - Este paquete contiene las clases contenedoras de nivel superior para la aplicación JavaFX.
* javafx.scene - Este paquete proporciona clases e interfaces para soportar el gráfico de la escena. Además, también proporciona subpaquetes como canvas, chart, control, effect, image, input, layout, media, paint, shape, text, transform, web, etc. Hay varios componentes que soportan esta rica API de JavaFX.



### Gráfico de Escena (Scene Graph)

En JavaFX, las Aplicaciones GUI fueron codificadas usando un Gráfico de Escena. Un Gráfico de Escena es el punto de partida de la construcción de la Aplicación GUI. Contiene las primitivas de la aplicación (GUI) que se denominan nodos.

Un nodo es un objeto visual/gráfico y puede incluir

* Objetos geométricos (gráficos) - (2D y 3D) como círculo, rectángulo, polígono, etc.
* Controles de interfaz de usuario - como botón, casilla de verificación, cuadro de elección, área de texto, etc.
* Contenedores - (paneles de diseño) como panel de bordes, panel de cuadrícula, panel de flujo, etc.
* Elementos multimedia - como objetos de audio, vídeo e imagen.

En general, una colección de nodos constituye un gráfico de escena. Todos estos nodos están dispuestos en un orden jerárquico como se muestra a continuación.

Cada nodo del gráfico de la escena tiene un único padre, y el nodo que no contiene ningún padre se conoce como nodo raíz.

Del mismo modo, cada nodo tiene uno o más hijos, y el nodo sin hijos se denomina nodo hoja; un nodo con hijos se denomina nodo rama.

Una instancia de nodo sólo puede añadirse a un gráfico de escena una vez. Los nodos de un gráfico de escena pueden tener Efectos, Opacidad, Transformaciones, Manejadores de eventos, Estados específicos de la aplicación.

### Quantum Toolkit

Es una abstracción sobre los componentes de bajo nivel de Prism, Glass, Media Engine y Web Engine. Une Prism y GWT y los pone a disposición de JavaFX.

### Prisma (Prism)

Prism es un pipeline gráfico de alto rendimiento acelerado por hardware que se utiliza para renderizar los gráficos en JavaFX. Puede renderizar gráficos tanto en 2D como en 3D.

Para renderizar los gráficos, Prism utiliza

* DirectX 9 en Windows XP y Vista.
* DirectX 11 en Windows 7.
* OpenGL en Linux, Sistemas Embebidos, entre otros.

Cuando se utiliza con una tarjeta gráfica o GPU compatible, ofrece gráficos más fluidos. En caso de que el sistema no soporte una tarjeta gráfica, entonces Prism utiliza por defecto la pila de renderizado por software.

### GWT (Glass Windowing Toolkit)

Como su nombre indica, GWT proporciona servicios para gestionar ventanas, temporizadores, superficies y colas de eventos. GWT conecta la plataforma JavaFX con el sistema operativo nativo.

### Vista Web (WebView)

Utilizando JavaFX, también se puede incrustar contenido HTML en un gráfico de escena. WebView es el componente de JavaFX que se utiliza para procesar este contenido. Utiliza una tecnología llamada Web Kit, que es un motor interno de navegador web de código abierto. Este componente soporta diferentes tecnologías web como HTML5, CSS, JavaScript, DOM y SVG.

Con WebView, usted puede:

* Renderizar contenido HTML desde una URL local o remota.
* Soportar el historial y proporcionar navegación hacia atrás y hacia adelante.
* Recargar el contenido.
* Aplicar efectos al componente web.
* Editar el contenido HTML.
* Ejecutar comandos de JavaScript.
* Manejar eventos.

En general, usando WebView, puedes controlar el contenido web desde Java.

### Motor de medios (Media Engine)

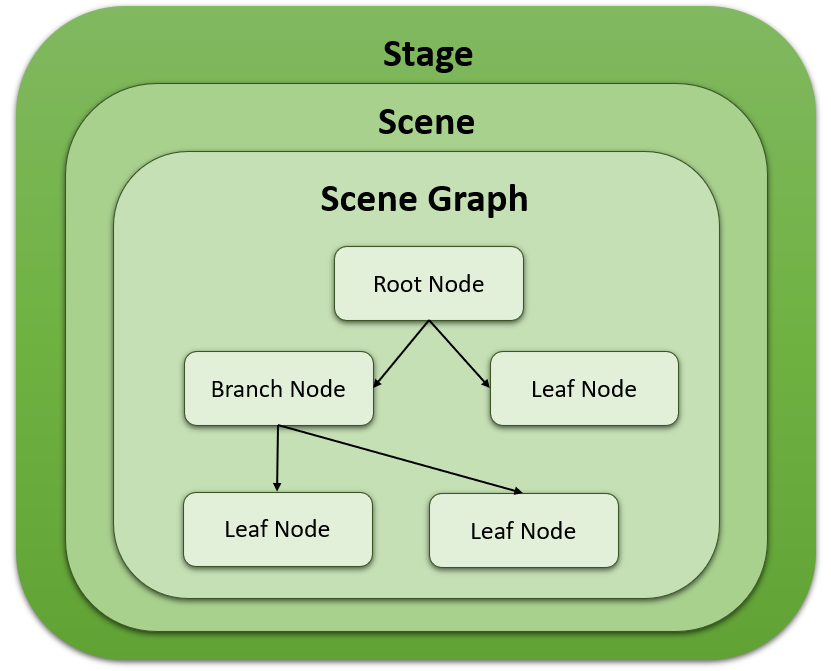
El motor multimedia de JavaFX se basa en un motor de código abierto conocido como Streamer. Este motor multimedia admite la reproducción de contenidos de vídeo y audio.

El paquete javafx.scene.media contiene las clases e interfaces para proporcionar la funcionalidad de los medios de comunicación en JavaFX. Se proporciona en forma de tres componentes, que son

* Media Object - Representa un archivo multimedia
* Media Player - Para reproducir contenido multimedia.
* Media View - Para mostrar los medios de comunicación.

# Estructura de la aplicación

En general, una aplicación JavaFX tendrá tres componentes principales: escenario, escena y nodos, como se muestra en el siguiente diagrama.



## Stage

Un stage (una ventana) contiene todos los objetos de una aplicación JavaFX. Está representado por la clase Stage del paquete javafx.stage. El stage primario es creado por la propia plataforma.

El objeto stage creado se pasa como argumento al método start() de la clase Application (explicado en la siguiente sección).

Un escenario tiene dos parámetros que determinan su posición: la anchura y la altura. Se divide en Área de Contenido y Decoraciones (Barra de Título y Bordes).

Hay cinco tipos de escenarios disponibles

* Decorado
* Sin decoración
* Transparente
* Unificado
* Utilidad

Hay que llamar al método show() para mostrar el contenido de un escenario.

## Escena (Scene)

Una escena representa el contenido físico de una aplicación JavaFX. Contiene todo el contenido de un gráfico de escena. La clase Scene del paquete javafx.scene representa el objeto scene. En una instancia, el objeto de escena se añade a un solo escenario.

Puedes crear una escena instanciando la clase Scene. Puedes optar por el tamaño de la escena pasando sus dimensiones (altura y anchura) junto con el nodo raíz a su constructor.

## Gráficos de escena y nodos (Scene Graph and Nodes)

Un gráfico de escena es una estructura de datos en forma de árbol (jerárquico) que representa el contenido de una escena. En cambio, un nodo es un objeto visual/gráfico de un gráfico de escena. Un nodo puede incluir:

* Objetos geométricos (gráficos) (2D y 3D) como - Círculo, Rectángulo, Polígono, etc.
* Controles de interfaz de usuario como - Botón, casilla de verificación, cuadro de elección, área de texto, etc.
* Contenedores (paneles de diseño) como panel de bordes, panel de cuadrícula, panel de flujo, etc.
* Elementos multimedia como objetos de audio, vídeo e imagen.

La clase Node del paquete javafx.scene representa un nodo en JavaFX, esta clase es la superclase de todos los nodos.

Como se ha comentado anteriormente un nodo es de tres tipos -

* Nodo Raíz - El primer gráfico de escena se conoce como nodo raíz.
* Nodo Rama/Nodo Padre - El nodo con nodos hijos se conoce como nodos rama/padre. La clase abstracta llamada Parent del paquete javafx.scene es la clase base de todos los nodos padre, y esos nodos padre serán de los siguientes tipos -
  + Grupo - Un nodo grupo es un nodo colectivo que contiene una lista de nodos hijos. Cada vez que se renderiza el nodo grupo, todos sus nodos hijos se renderizan en orden. Cualquier transformación, efecto de estado aplicado en el grupo se aplicará a todos los nodos hijos.
  + Region - Es la clase base de todos los controles de interfaz de usuario basados en JavaFX Node, como Chart, Pane y Control.
  + WebView - Este nodo gestiona el motor web y muestra su contenido.
* Nodo Hoja - El nodo sin nodos hijos se conoce como nodo hoja. Por ejemplo, Rectángulo, Elipse, Caja, ImageView, MediaView son ejemplos de nodos hoja.

Es obligatorio pasar el nodo raíz al gráfico de la escena. Si se pasa el Grupo como raíz, todos los nodos se recortarán a la escena y cualquier alteración en el tamaño de la escena no afectará al diseño de este.

## Ejercicio: Primera aplicación JavaFX

“Example 1 – Creating an Empty Window” de

https://www.tutorialspoint.com/javafx/javafx\_application.htm

# Controles de UI

Toda interfaz de usuario tiene en cuenta los tres aspectos principales siguientes:

* Elementos de la interfaz de usuario - Son los elementos visuales principales que el usuario acaba viendo y con los que interactúa. JavaFX proporciona una enorme lista de elementos comunes y ampliamente utilizados que varían desde los más básicos hasta los más complejos.
* Layouts - Definen cómo deben organizarse los elementos de la UI en la pantalla y proporcionan un aspecto final a la GUI (Graphical User Interface).
* Comportamiento - Son eventos que ocurren cuando el usuario interactúa con los elementos de la UI.

## Elementos UI

JavaFX proporciona varias clases en el paquete javafx.scene.control. Para crear diversos componentes de la interfaz gráfica de usuario (controles), JavaFX admite varios controles, como el selector de fecha, el campo de texto del botón, etc.

Cada control está representado por una clase; se puede crear un control instanciando su respectiva clase.

|  |  |
| --- | --- |
| **Control** | **Descripción** |
| Label | Un objeto Label es un componente para colocar texto. |
| Button | Esta clase crea un botón etiquetado. |
| ColorPicker | Un ColorPicker proporciona un panel de controles diseñado para permitir al usuario manipular y seleccionar un color. |
| CheckBox | Un CheckBox es un componente gráfico que puede estar en estado on (verdadero) o off (falso). |
| RadioButton | La clase RadioButton es un componente gráfico que puede estar en estado ON (verdadero) u OFF (falso) en un grupo. |
| ListView | Un componente ListView presenta al usuario una lista desplazable de elementos de texto. |
| TextField | Un objeto TextField es un componente de texto que permite la edición de una sola línea de texto. |
| PasswordField | Un objeto PasswordField es un componente de texto especializado en la introducción de contraseñas. |
| Scrollbar | Un control Scrollbar representa un componente de barra de desplazamiento para permitir al usuario seleccionar entre un rango de valores. |
| FileChooser | Un control FileChooser representa una ventana de diálogo en la que el usuario puede seleccionar un archivo. |
| ProgressBar | A medida que la tarea avanza hacia su finalización, la barra de progreso muestra el porcentaje de finalización de la tarea. |
| Slider | Un deslizador permite al usuario seleccionar gráficamente un valor deslizando un mando dentro de un intervalo delimitado. |

## Layouts

Después de construir todos los nodos necesarios en una escena, generalmente los colocaremos en orden.

Esta disposición de los componentes dentro del contenedor se denomina Disposición del contenedor. También podemos decir que hemos seguido un layout ya que incluye la colocación de todos los componentes en una posición determinada dentro del contenedor.

JavaFX proporciona varios layouts predefinidos como HBox, VBox, Border Pane, Stack Pane, Text Flow, Anchor Pane, Tile Pane, Grid Pane, Flow Panel, etc., todas representado por una clase y todas estas clases pertenecen al paquete javafx.layout. La clase llamada Pane es la clase base de todos los layouts en JavaFX.

* HBox

El layout HBox organiza todos los nodos de nuestra aplicación en una única fila horizontal. La clase llamada HBox del paquete javafx.scene.layout representa el layout de caja horizontal de texto.

* VBox

El layout VBox organiza todos los nodos de nuestra aplicación en una sola columna vertical. La clase llamada VBox del paquete javafx.scene.layout representa el layout de caja vertical de texto.

* BorderPane

El diseño del Border Pane organiza los nodos de nuestra aplicación en las posiciones superior, izquierda, derecha, inferior y central. La clase llamada BorderPane del paquete javafx.scene.layout representa el layout del border pane.

* StackPane

El diseño del panel de la pila organiza los nodos de nuestra aplicación uno encima de otro como en una pila. El nodo añadido primero se coloca en la parte inferior de la pila y el siguiente nodo se coloca encima de él. La clase StackPane del paquete javafx.scene.layout representa el diseño del panel de pila.

* TextFlow

El diseño de flujo de texto organiza múltiples nodos de texto en un solo flujo. La clase denominada TextFlow del paquete javafx.scene.layout representa el diseño de flujo de texto.

* AnchorPane

El diseño del Anchor Pane ancla los nodos de nuestra aplicación a una distancia determinada del panel. La clase denominada AnchorPane del paquete javafx.scene.layout representa el diseño del Anchor Pane.

* TilePane

El layout Tile Pane añade todos los nodos de nuestra aplicación en forma de tiles de tamaño uniforme. La clase denominada TilePane del paquete javafx.scene.layout representa el layout TilePane.

* GridPane

El diseño del panel de cuadrícula organiza los nodos de nuestra aplicación como una cuadrícula de filas y columnas. Este diseño es muy útil para crear formularios con JavaFX. La clase GridPane del paquete javafx.scene.layout representa el diseño GridPane.

* FlowPane

El diseño del panel de flujo envuelve todos los nodos de un flujo. Un panel de flujo horizontal envuelve los elementos del panel en su altura, mientras que un panel de flujo vertical envuelve los elementos en su anchura. La clase denominada FlowPane del paquete javafx.scene.layout representa el diseño del panel de flujo.

## Comportamiento

En JavaFX, podemos desarrollar aplicaciones GUI, aplicaciones web y aplicaciones gráficas. En estas aplicaciones, cada vez que un usuario interactúa con la aplicación (nodos), se dice que se ha producido un evento.

Por ejemplo, hacer clic en un botón, mover el ratón, introducir un carácter a través del teclado, seleccionar un elemento de la lista, desplazar la página son las actividades que hacen que se produzca un evento.

### Tipos de eventos

Los eventos pueden clasificarse a grandes rasgos en las dos categorías siguientes

* Eventos en primer plano - Aquellos eventos que requieren la interacción directa de un usuario. Se generan como consecuencia de la interacción de una persona con los componentes gráficos de una interfaz gráfica de usuario. Por ejemplo, hacer clic en un botón, mover el ratón, introducir un carácter a través del teclado, seleccionar un elemento de la lista, desplazar la página, etc.
* Eventos de fondo - Los eventos que requieren la interacción del usuario final se conocen como eventos de fondo. Las interrupciones del sistema operativo, los fallos de hardware o software, la expiración de un temporizador o la finalización de una operación son ejemplos de eventos en segundo plano.

### Eventos

JavaFX proporciona soporte para manejar una amplia variedad de eventos. La clase Event del paquete javafx.event es la clase base de un evento.

Una instancia de cualquiera de sus subclases es un evento. JavaFX proporciona una gran variedad de eventos. Algunos de ellos se enumeran a continuación.

* Evento de ratón - Es un evento de entrada que ocurre cuando se hace clic con el ratón. Está representado por la clase llamada MouseEvent. Incluye acciones como el clic del ratón, la pulsación del ratón, la liberación del ratón, el movimiento del ratón, la entrada del ratón en el objetivo, la salida del ratón en el objetivo, etc.
* Evento de Tecla - Es un evento de entrada que indica la pulsación de una tecla en un nodo. Está representado por la clase llamada KeyEvent. Este evento incluye acciones como tecla pulsada, tecla liberada y tecla tecleada.
* Evento de Arrastre - Este es un evento de entrada que ocurre cuando el ratón es arrastrado. Está representado por la clase llamada DragEvent. Incluye acciones como arrastrar introducido, arrastrar soltado, arrastrar objetivo introducido, arrastrar objetivo salido, arrastrar sobre, etc.
* Evento de Ventana - Este es un evento relacionado con las acciones de mostrar/ocultar la ventana. Está representado por la clase llamada WindowEvent. Incluye acciones como ocultar ventana, mostrar ventana, ocultar ventana, mostrar ventana, etc.

# Referencias

<https://www.tutorialspoint.com/javafx/index.htm>

<https://o7planning.org/11009/javafx>