
JavaFX™

Tabla de contenido:

Nota: Es una simple y sencilla traducción del tutorial que se encuentra en la página oficial de oracle sobre los primeros pasos en JavaFX.

Fecha de traducción 1-Abril-2014.

No me hago responsable del mal uso de este documento, todas las marcas, herramientas y empresas mencionadas quedan fuera de mi responsabilidad.

Norman Carcamo.

Prefacio

En este prólogo se da una visión general de este tutorial y también describe las características de accesibilidad de documentos y convenciones utilizadas en este tutorial - *Conceptos básicos del JavaFX*

Acerca de este tutorial

Este tutorial es una recopilación de tres documentos que fueron entregados previamente con el conjunto de la documentación 2.x JavaFX: JavaFX información general(Overview), Arquitectura de JavaFX (Understanding the JavaFX Architecture) y Primeros pasos con JavaFX (Getting Started with JavaFX). El contenido combinado se ha mejorado con información actualizada sobre las nuevas características de JavaFX incluido con el lanzamiento de Java SE 8. Este documento contiene las siguientes partes:

- ¿Que es JavaFX?
- Primeros pasos con JavaFX Aplicaciones de ejemplo

Cada parte contiene los capítulos que introducen a la tecnología JavaFX, y le ayudará a comenzar a aprender cómo usarlo para su desarrollo de aplicaciones.

Audiencia

Este documento está dirigido a los desarrolladores de JavaFX.

Accesibilidad Documentación

Para obtener información sobre el compromiso de Oracle con la accesibilidad, visite el sitio web del Programa de Accesibilidad de Oracle en http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Acceso a Soporte Oracle

Los clientes de Oracle tienen acceso a soporte electrónico a través de My Oracle Support. Para obtener más información, visite http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info o visite http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs si con problemas de audición.

Documentos relacionados

Para obtener más información, consulte el resto de la documentación JavaFX puesta en http://docs.oracle.com/javase/javase-clienttechnologies.htm.

Convenciones

Las siguientes convenciones de texto se utilizan en este documento:

Convención	Significado
boldface	La negrita indica elementos de la interfaz gráfica de usuario asociado a una acción, o términos definidos en el texto o en el glosario.
italic	El texto en cursiva indica títulos de libros, énfasis, o variables de marcador de posición para la cual usted provee valores particulares.
monospace	El monoespaciado indica mandatos dentro de un párrafo, las direcciones URL, el código en los ejemplos, el texto que aparece en la pantalla, o el texto que introduzca.

Novedades

Este capítulo resume las nuevas características y cambios importantes realizados en los componentes de productos de JavaFX y de la liberación de Java SE 8.

- El nuevo tema de Módena es ahora el tema por defecto para las aplicaciones JavaFX. Vea la sección del tema Modena en características clave.

- Se ha añadido soporte para las funciones adicionales de HTML5. Consulte Cómo agregar contenido HTML para aplicaciones JavaFX para más información.

- La nueva clase SwingNode mejora la función de interoperabilidad Swing. Consulte Incorporación de contenido Swing en aplicaciones JavaFX.

- Nuevos controles integrados de Interfaz de usuario, DatePicker y TableView ya están disponibles. Consulte el documento Uso de Controles de interfaz de usuario JavaFX para más información.

- La biblioteca 3D de gráficos ha sido mejorada con varias clases de API nuevas. Ver Gráficos 3D con la sección de Características principales , y Introducción a JavaFX Gráficos 3D para más información.

- El paquete javafx.print ya está disponible y proporciona las APIs JavaFX publicas de impresión.

- Se ha añadido soporte de texto enriquecido.

- Compatibilidad con pantallas de alta DPI se han puesto a disposición.

- Clases styleables CSS se convirtieron en API públicas.

- Clase de servicio programado(Scheduled) se ha introducido.

¿Que es JavaFX?

Parte 1 contiene los siguientes capitulos:

- JavaFX Información General(JavaFX Overview)
- Entendiendo la Arquitectura JavaFX (Understanding the JavaFX Architecture)

JavaFX Información General

Este capítulo proporciona una visión general de los tipos de aplicaciones que se pueden construir utilizando las API de JavaFX, dónde descargar las librerías de JavaFX, y una información de alto nivel acerca de JavaFX y las características principales que está entregando.

JavaFX es un conjunto de gráficos y paquetes de comunicación que permite a los desarrolladores para diseñar, crear, probar, depurar y desplegar aplicaciones cliente enriquecidas que operan constantemente a través de diversas plataformas.

- Aplicaciones JavaFX
- disponibilidad
- Características principales
- ¿Qué puedo construir con JavaFX?
- ¿Cómo se ejecuta una la aplicación de ejemplo?
- ¿Cómo se ejecuta una muestra en un IDE?
- ¿Cómo se crea una aplicación JavaFX?
- Recursos

Consulte Entendiendo la Arquitectura JavaFX acerca de la arquitectura de la plataforma JavaFX y para obtener una breve descripción de las API de JavaFX para streaming, renderizado web y la estilización de interfaz de usuario con CSS.

Aplicaciones JavaFX

La biblioteca de JavaFX está escrita como una API de Java, las aplicaciones JavaFX puede hacer referencia a APIs de código de cualquier biblioteca Java. Por ejemplo, las aplicaciones JavaFX pueden utilizar las bibliotecas de API de Java para acceder a las capacidades del sistema nativas y conectarse a aplicaciones de middleware basadas en servidor.

La apariencia de las aplicaciones JavaFX se pueden personalizar. Las Hojas de Estilo en Cascada (CSS) separan la apariencia y estilo de la logica de la aplicación para que los desarrolladores puedan concentrarse en el código. Los diseñadores gráficos pueden personalizar fácilmente el aspecto y el estilo de la aplicación a

través de CSS. Si usted tiene un diseño de fondo de la web, o si usted desea separar la interfaz de usuario (UI) y la lógica de servidor, entonces, usted puede desarrollar los aspectos de la presentación de la interfaz de usuario en el lenguaje de scripting FXML y use el código de Java para la aplicación lógica. Si prefiere diseñar interfaces de usuario sin necesidad de escribir código, entonces, utilice JavaFX Scene Builder. Al diseñar la interfaz de usuario con javaFX Scene Builder el crea código de marcado FXML que pueden ser portado a un entorno de desarrollo integrado (IDE) de forma que los desarrolladores pueden añadir la lógica de negocio.

Disponibilidad

Las API de JavaFX están disponibles como una característica totalmente integrada del Java SE Runtime Environment (JRE) , y Java Development Kit (JDK). Debido a que el JDK se encuentra disponible para todas las principales plataformas de escritorio (Windows, Mac OS X y Linux), aplicaciones JavaFX compiladas en el JDK 7 y posteriormente se ejecutan también en todas las principales plataformas de escritorio. Soporte para plataformas ARM también ha sido puesto a disposición del publico con JavaFX 8. JDK para ARM Incluye la base de gráficos y controles componentes de JavaFX.

La compatibilidad entre plataformas permite una experiencia de ejecución consistente para los desarrolladores de aplicaciones JavaFX y usuarios. Oracle asegura versiones sincronizadas y actualizaciones en todas las plataformas y ofrece un amplio programa de apoyo a las empresas que ejecutan aplicaciones de misión crítica.

En la página de descarga de JDK, puede obtener un archivo zip de ejemplos de aplicaciones JavaFX. Las aplicaciones de ejemplo ofrecen muchos ejemplos de código y los fragmentos que muestran por ejemplo la forma de escribir aplicaciones JavaFX. Consulte "¿Cómo puedo ejecutar una aplicación de ejemplo?" para más información.

Caracteristicas Claves

Las siguientes características se incluyen en JavaFX 8 y versiones posteriores. Los elementos que se introdujeron en el lanzamiento JavaFX 8 se indican a continuación:

- **APIs de Java**. JavaFX es una biblioteca Java que consiste en clases e interfaces que están escritas en código Java. Las APIs están diseñadas para ser una alternativa amigable con la máquina virtual de Java (Java VM) como los lenguajes JRuby, y Scala.

- **FXML y Scene Builder**. FXML es un lenguaje de marcado declarativo basado en XML para la construcción de una interfaz de usuario de aplicaciones JavaFX. Un diseñador puede codificar en FXML o utilizar JavaFX Scene Builder para diseñar de forma interactiva la interfaz gráfica de usuario (GUI). Scene Builder

genera marcado FXML que pueden ser portado a un IDE para que un desarrollador pueda añadir la lógica de negocio.

- **WebView**. Un componente web que utiliza la tecnología WebKitHTML para hacer posible incrustar(embeber), páginas web dentro de una aplicación JavaFX. JavaScript se ejecuta dentro de un WebView y se puede llamar a las APIs de Java y las APIs de Java pueden llamar a JavaScript que se ejecuta dentro de un WebView. Compatibilidad con características adicionales de HTML5, incluyendo Web Sockets, Web Workers y fuentes Web, y capacidades de impresión se han añadido en JavaFX 8. Consulte Cómo agregar contenido HTML en aplicaciones JavaFX.

- **Interoperabilidad Swing**. Las aplicaciones Swing existentes pueden ser actualizadas con las características de JavaFX, como gráficos multimedia enriquecidos de reproducción y el contenido web incorporado. La clase SwingNode, que le permite incluir contenido dentro de aplicaciones JavaFX se ha añadido en JavaFX 8. Consulte el Javadoc API de SwingNode e Incrustación de contenido Swing en aplicaciones JavaFX para más información.

- **Controles de interfaz de usuario(UI) incorporados(Built-in) y CSS**. JavaFX ofrece todos los principales controles UI que se requieren para desarrollar una aplicación con todas las caracteristicas necesarias. Los componentes pueden ser decorados con las tecnologías Web estándar como CSS. Los controles DatePicker y TreeTableView UI están ahora disponibles con el lanzamiento de JavaFX 8. Consulte Uso de controles UI en JavaFX para más información. Además las clases CSS Styleables* se han convertido en APIs públicas, permitiendo que los objetos sean de estilo de CSS.

- **Tema de Módena**. El tema Modena reemplaza el tema Caspio como valor predeterminado para aplicaciones JavaFX 8. El tema del Caspio está disponible para su uso aún mediante la adición de la línea setUserAgentStylesheet(STYLESHEET_CASPIAN) en su método start() de la aplicación. Para obtener más información, consulte el blog Modena en fxexperience.com

- **Características de Gráficos 3D**. Las nuevas clases de la API para Shape3D (Box, Cylinder, MeshView y subclases de Sphere), SubScene, Material, PickResult, LightBase (AmbientLight , y subclases de PointLight) y SceneAntialiasing se han añadido a la biblioteca de gráficos 3D en JavaFX 8. La API de la clase Camera también se ha actualizada en esta versión. Para obtener más información, consulte el documento Guía de inicio JavaFX gráficos 3D, y el javadoc API correspondiente para javafx.scene.shape.Shape3D, javafx.scene.SubScene, javafx.scene.paint.Material, javafx.scene.input.PickResult y javafx.scene.SceneAntialiasing.

- **API Canvas**. La API de Canvas permite dibujar directamente dentro de un área de la scene JavaFX que consta de un elemento gráfico (nodo).

- **API de impresión**. El paquete javafx.print se ha añadido en el lanzamiento Java SE 8 y proporciona clases públicas para la impresión con JavaFX.

- **Soporte de texto enriquecido**. JavaFX 8 trae soporte mejorado de texto para JavaFX, incluyendo texto bidireccional, y scripts de texto complejos, como el tailandés e hindú en los controles y de varias líneas, texto multi-estilo en los nodos de texto.

- **Soporte multitouch**. JavaFX proporciona apoyo a las operaciones multitouch, con base en las capacidades de la plataforma subyacente.

- **Soporte Hi-DPI**. JavaFX 8 es ahora compatible con pantallas de alta DPI.

- **Aceleración por hardware con canalización de gráficos**. Los gráficos JavaFX se basan en el canal de renderizado de gráficos (el hilo Prism). JavaFX ofrece gráficos suaves(smooth) que hacen que rápidamente a través el hilo Prism cuando se utiliza con una tarjeta de gráficos compatible o unidad de procesamiento de gráficos (GPU). Si un sistema no dispone de una de las GPUs recomendadas soportados por JavaFX, recibe el valor predeterminado el hilo Prism a la pila de software de renderización.

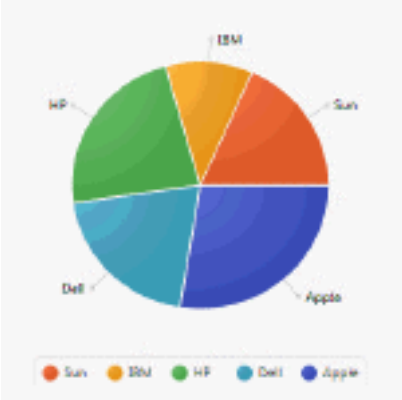
- **Motor de medios de alto rendimiento.** El oleoducto multimedia es compatible con la reproducción de contenido multimedia en la Web. Proporciona un marco multimedia estable, de baja latencia que se basa en el framework multimedia GStreamer.

- **Modelo de implementación de aplicaciones independiente del S.O**. Paquetes de aplicaciones autónomas tienen todos los recursos de la aplicación, y una copia privada de los tiempos de ejecución de Java y JavaFX. Se distribuyen en forma de paquetes instalables nativos como por ejemplo .dmg, .deb .exe .etc. y proporcionan la misma instalación y la experiencia puesta en marcha como aplicaciones nativas para ese sistema operativo.

¿Qué puedo construir una aplicación con JavaFX?

Con JavaFX puedes construir muchos tipos de aplicaciones. Por lo general, son las aplicaciones de red y estas que se implementan a través de múltiples plataformas y mostrar información en una interfaz de usuario moderna de alto rendimiento que cuenta con audio, vídeo, gráficos y animación.

Tabla 1-1 muestra las imágenes de algunas de las aplicaciones de ejemplo de JavaFX que se incluyeron con la versión 8.n de JavaFX.

Aplicación de Ejemplo	Descripción
 Descripción de la ilustración ensemble-small.gif	**JavaFX Ensemble8** Ensemble8 es una galería de aplicaciones de ejemplo que muestran una gran variedad de funciones de JavaFX, incluyendo la animación, gráficos, y los controles. Usted puede ver e interactuar con cada muestra que se ejecuta en todas las plataformas, y leer sus descripciones. En las plataformas de escritorio, puede copiar el código fuente de cada muestra, ajustar las propiedades de los componentes de la muestra se utilizan en varias muestras, y siga los enlaces a la documentación de la API relevante cuando se está conectado a la Internet. Ensemble8 también corre con JavaFX para ARM.
 Descripción de la ilustración modena-sample.gif	**Modena** Módena es una aplicación de ejemplo que muestra la apariencia de los componentes UI utilizando el tema de Módena. Te da la opción de contrastar los temas Módena y Caspio, y explorar diversos aspectos de estos temas.

Aplicación de Ejemplo	Descripción
 Description of the illustration sample-3dviewer.gif	**3D Viewer** Es una aplicación de ejemplo que le permite navegar y examinar una escena 3D con un ratón o un trackpad. 3D Viewer tiene importadores para un subconjunto de las funciones de archivos OBJ y maya. También se prevé la posibilidad de importar archivos de animación maya. (Tenga en cuenta que en el caso de los archivos de Maya, el historial de la construcción debería suprimirse en todos los objetos cuando se guarda como un archivo de Maya.) 3dviewer también tiene la capacidad de exportar el contenido de la escena como archivos Java o FXML.

¿Cómo se ejecuta una la aplicación de ejemplo?

Los pasos de esta sección se explica cómo descargar y ejecutar las aplicaciones de muestra que están disponibles como descarga independiente en la Plataforma Java (JDK 8).

Nota: Antes de poder ejecutar una aplicación JavaFX de muestra, es necesario tener las bibliotecas de tiempo de ejecución de JavaFX en su máquina. Antes de continuar con estos pasos, instale la última versión del JDK 8 o la última versión del JRE.

Para descargar y ejecutar las aplicaciones de ejemplo:

1. Ir a la página de Java SE Descargas en http://www.oracle.com/technetwork/java/javase/downloads/index.html

2. Desplácese hacia abajo y localice la sección Muestras JDK 8 y JavaFX Demos.

3. Haga clic en los Ejemplos y demostraciones para ir a la página de descargas.

4. En la página de Java SE Development Kit 8 Descargas, desplácese hacia abajo para JavaFX Demos y Muestras en la sección de Descargas.

5. Descargue el archivo zip para el sistema operativo correcto y extraer los archivos.

   Se crea el directorio de JavaFX-muestras-8.x , y contiene los archivos para las muestras disponibles. Los proyectos de NetBeans para las muestras están en el directorio \ src JavaFX-muestras-8.x.

6. Haga doble clic en el archivo ejecutable para una muestra.

   Por ejemplo, para ejecutar la aplicación de ejemplo pre-construidos Ensemble 8, haga doble clic en el archivo 8.jar Ensemble.

¿Cómo se ejecuta una muestra en un IDE?

Puede utilizar varios IDEs de desarrollo Java para desarrollar aplicaciones JavaFX. Los siguientes pasos explican cómo ver y ejecutar el código fuente en el IDE NetBeans.

Para ver y ejecutar el código fuente de ejemplo en el IDE NetBeans:

1. Descargar las muestras como se describe anteriormente y extraer los archivos.

2. Desde el NetBeans 7.4 o posterior IDE, cargue el proyecto para el ejemplo que desea ver.

   a. En el menú **Archivo**, seleccione **Abrir proyecto**.

   b. En el cuadro de diálogo **Abrir proyecto**, vaya al directorio que enumera las muestras. La ruta de navegación se ve algo como esto:

   ..\javafx_samples-8.x-<platform>\javafx-samples-8.x\src

   c. Seleccione la muestra que desea ver.

   d. Haga clic en el botón **Abrir proyecto**.

   e. En la ventana Proyectos, haga clic derecho en el proyecto que acaba de abrir y seleccione **Ejecutar.**

3. Observe la ventana de resultados se actualiza y el proyecto de ejemplo se ejecuta , y despliega.

¿Cómo se crea una aplicación JavaFX?

Debido a que las aplicaciones JavaFX están escritos en el lenguaje Java, puede utilizar su editor favorito o cualquier entorno de desarrollo integrado (IDE) que admita el lenguaje Java (como NetBeans, Eclipse o IntelliJ IDEA) para crear aplicaciones JavaFX.

Para crear aplicaciones JavaFX:

1. **Ir a la página de Java SE Descargas en** http://www.oracle.com/technetwork/java/javase/downloads/index.html para descargar el JDK de Oracle ® 8 con soporte 8.n JavaFX. Enlaces a las configuraciones de sistemas certificados y notas de la versión también están disponibles en esa página…

2. Utilice **Primeros pasos con JavaFX Aplicaciones** de ejemplo para crear aplicaciones simples que muestra cómo trabajar con presentaciones, hojas de estilo y los efectos visuales.

3. Utilice **JavaFX Scene Builder** para diseñar la interfaz de usuario para su aplicación JavaFX sin necesidad de programación. Puede arrastrar y soltar los componentes UI a un área de trabajo, modificar sus propiedades, solicitar las hojas de estilo, e integrar el código resultante con la lógica de la aplicación.
    a. Descargue JavaFX Scene Builder desde la página de Descargas de JavaFX http://www.oracle.com/technetwork/java/javase/downloads/index.html
    b. Siga la Guía tutorial de introducción a JavaFX Scene Builder para aprender más.

Recursos

Utilice los siguientes recursos para aprender más acerca de la tecnología JavaFX.

Descargue la última liberación del JDK 8 y las muestras de JavaFX desde la página de descargas de Java SE en: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Leer Entender la arquitectura del JavaFX.

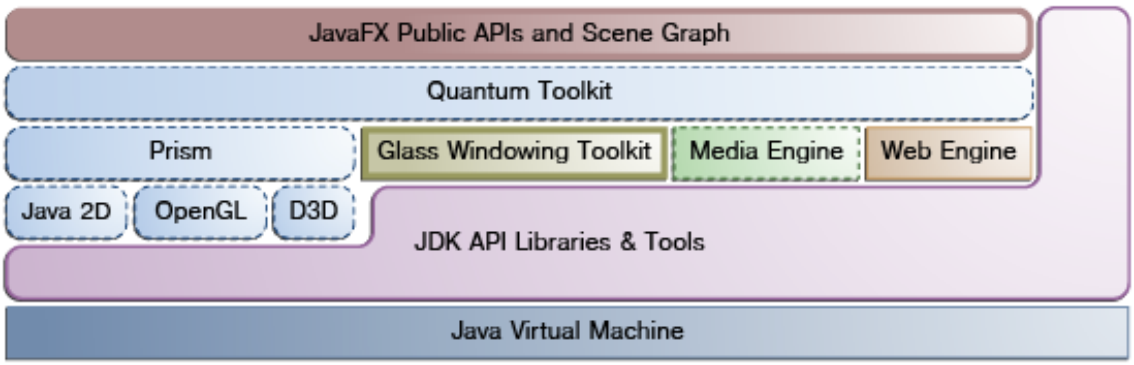
Examine tutoriales y artículos JavaFX para desarrolladores.

Entendiendo la Arquitectura JavaFX

Figura 2-1 ilustra los componentes de la arquitectura de la plataforma JavaFX. Las secciones siguientes describen el diagrama de cada componente y cómo la interconexión de las partes. Por debajo de las APIs públicas JavaFX se encuentra el motor que funciona su código JavaFX. Se compone de subcomponentes que incluyen un motor de JavaFX gráficos de alto rendimiento, llamado Prism; un sistema de ventanas pequeño y eficiente, llamado Glass; un motor de multimedia, y un motor web. Aunque estos componentes no están expuestos públicamente, sus descripciones pueden ayudar a entender mejor lo que se ejecuta una aplicación JavaFX.

- Scene Graph
- Java Public APIs for JavaFX Features
- Graphics System
- Glass Windowing Toolkit
- Media and Images
- Web Component
- CSS
- UI Controls
- Layout
- 2-D and 3-D Transformations
- Visual Effects

Figure 2-1 JavaFX Architecture Diagram



El gráfico de la escena(Scene Graph) JavaFX, que se muestra como parte de la capa superior de la figura 2-1, es el punto de partida para la construcción de una aplicación JavaFX. Es un árbol jerárquico de nodos que representa todos los elementos visuales de la interfaz de usuario de la aplicación. Puede controlar la entrada y puede ser entregado.

Un solo elemento en un escenario gráfico(Scene Graph) se denomina nodo. Cada nodo tiene un ID, clase de estilo, y el volumen de delimitación. Con la excepción del nodo raíz de un gráfico de la escena, cada nodo en un gráfico de la escena tiene un solo padre y cero o más hijos. También puede tener los siguientes:

- Efectos, como desenfoques(blurs) y sombras(shadows)
- opacidad (Opacity)
- Transformaciones (Transforms)
- Los controladores de eventos (como el ratón, llave y método de entrada)
  (Event handlers (such as mouse, key and input method))
- Un estado de aplicación específica(An application-specific state)

A diferencia de Swing y Abstract Window Toolkit (AWT), el gráfico de la escena JavaFX incluye también las primitivas gráficas, como rectángulos(Rectangles) y texto(Text), además de tener controles(Controls), contenedores de diseño (Layout Containers), imágenes(Images) y multimedia(Media).

Para la mayoría de usos, el escenario gráfico(Scene Graph) simplifica el trabajo con las interfaces de usuario, especialmente cuando se utilizan interfaces de usuario ricas. Animación de varios gráficos en el escenario gráfico se puede lograr rápidamente utilizando las API javafx.animation y métodos declarativos, tales como doc XML, también funcionan bien.

La API javafx.scene permite la creación y especificación de varios tipos de contenido, tales como:

Nodos(Nodes): formas(Shapes) (2-D y 3-D), imágenes(Images), medios(Media), navegador web incorporado(embedded web browser), el texto(Text), los controles de interfaz de usuario(Controls UI), gráficos(Charts), grupos(Groups) y contenedores(containers).

Estado(State): Transformaciones(Transforms) (posicionamiento y orientación de los nodos), efectos visuales, y otro estado visual del contenido

Efectos(effects): objetos simples que cambian la apariencia de de nodos de gráficos de escena(Scene Graph), como los desenfoques(blurs), sombras(shadows) y ajuste de color(color adjustment).

Para obtener más información, consulte el Trabajo con JavaFX Scene Graph.

APIs Java públicas para caracteristicas JavaFX

La capa superior de la arquitectura de JavaFX se muestra en la Figura 2-1 proporciona un conjunto completo de API públicas de Java que apoyan aplicaciones de cliente enriquecido. Estas API proporcionan libertad y flexibilidad para construir aplicaciones de cliente enriquecido sin precedentes. La plataforma JavaFX combina las mejores capacidades de la plataforma Java con funcionalidad multimedia completa e inmersiva en un entorno de desarrollo de una ventanilla intuitiva y completa. Estos APIS de Java para las funciones JavaFX:

- Permitir el uso de potentes características de Java, como los genéricos(Generics), anotaciones(Annotations), multithreading y expresiones Lamda(Lambda expression, introducidas en Java SE 8).

- Que sea más fácil para los desarrolladores web para utilizar JavaFX de otros lenguajes dinámicos basados en JVM, como Groovy y JavaScript.

- Permitir a los desarrolladores de Java utilizar otros lenguajes del sistema, como Groovy, para escribir aplicaciones JavaFX grandes y complejas.

- Permitir el uso de bindings(vinculantes) que incluye el apoyo a las expresiones perezosas vinculantes(lazy binding) de alto rendimiento, las expresiones de secuencia encuadernadas(bound sequence expressions), y reevaluación parcial(bind partial). Lenguajes alternativos (como Groovy) pueden usar esta biblioteca de enlace(binding) para introducir la sintaxis de unión(binding) similar a la de JavaFX Script.

- Extender las colecciones de la biblioteca de Java para incluir listas y mapas observables, que permiten a las aplicaciones las interfaces de usuario los cables a los modelos de datos, observar los cambios en esos modelos de datos, y actualizar el control de la interfaz de usuario correspondiente en consecuencia.

Las API de JavaFX y el modelo de programación son una continuación de la línea de productos 1.x JavaFX. La mayoría de las API de JavaFX se han portado directamente a Java. Algunas API, tales como Diseño y Media junto con muchos otros detalles, se han mejorado y simplificado, basado en los comentarios recibidos de los usuarios de la versión 1.x de JavaFX. JavaFX se basa más en los estándares web, como CSS para los

controles de estilo y ARIA para las especificaciones de accesibilidad. El uso de estándares web adicionales también está bajo revisión.

Gráficos del Sistema(Graphics System)

El sistema de gráficos JavaFX, que se muestra en azul en la Figura 2-1, es un detalle de implementación debajo de la capa gráfica escena JavaFX. Es compatible tanto con 2-D y escenarios gráficos en 3-D. Proporciona la representación del software cuando el hardware de gráficos en un sistema es insuficiente para apoyar renderizado acelerado por hardware.

Dos ductos de aceleración de gráficos se implementan en la plataforma JavaFX:

- Los procesos de **Prism**. Puede funcionar en dos renderizadores de hardware y software, incluyendo 3-D. Es responsable de la rasterización y renderizado de escenas de JavaFX. A continuación múltiples caminos son posibles en función del dispositivo que se utiliza:
    - DirectX 9 en Windows XP y Windows Vista
    - DirectX 11 en Windows 7
    - OpenGL en Mac, Linux, Embedded
    - Software de representación cuando la aceleración de hardware no es posible
    - El hardware da vía de acceso completamente acelerado se utiliza siempre que sea posible, pero cuando no esté disponible, el software se utiliza path porque el software ya se distribuye en todos los entornos de ejecución Java (JRE). Esto es particularmente importante cuando manejan escenas 3-D. Sin embargo, el rendimiento es mejor cuando el hardware se utilizan estos path.
- Quantum Toolkit une Prism y Glass Windowing Toolkit juntos y los pone a disposición de la capa de JavaFX por encima de ellos en la pila. También gestiona las normas vinculadas a la prestación de enhebrado contra el control de eventos.

Kit de herramientas de ventanas Glass (Glass Windowing Toolkit)

El kit de herramientas de ventanas Glass, que se muestra en color beige en la parte central de la figura 2-1, es el nivel más bajo de la pila de gráficos JavaFX. Su responsabilidad principal es proporcionar servicios operativos nativos, tales como el manejo de las ventanas, temporizadores y superficies. Además, es la capa depende de la plataforma que conecta la plataforma JavaFX para el sistema operativo nativo.

El kit de herramientas Glass es también responsable de la gestión de la cola de eventos. A diferencia de Abstract Window Toolkit (AWT), que gestiona su propia cola de eventos, el kit de herramientas Glass utiliza la funcionalidad de cola de eventos del sistema operativo nativo para programar el uso de hilo. También a diferencia de AWT, el kit de herramientas Glass se ejecuta en el mismo subproceso que la aplicación JavaFX. En AWT, el medio natural de AWT se ejecuta en un hilo y el nivel de Java se ejecuta en otro hilo. Esto introduce una gran cantidad de temas, muchos de los cuales se resuelven en JavaFX utilizando el enfoque de subproceso de la aplicación única JavaFX.

Hilos(Threads)

El sistema ejecuta dos o más de los siguientes hilos en un momento dado:

- **JavaFX application thread**: Este es el hilo principal utilizado por los desarrolladores de aplicaciones JavaFX. Cualquier scene "en vivo", que es una escena que es parte de una ventana, se debe acceder desde este hilo. Un escenario gráfico se puede crear y manipular en un subproceso en segundo plano, pero cuando su nodo raíz se adjunta a cualquier objeto en vivo en la escena, ese escenario gráfico debe acceder desde el subproceso de la aplicación JavaFX. Esto permite a los desarrolladores crear gráficos de escenas complejas en un subproceso en segundo plano mientras se mantiene animaciones en escenas 'en vivo' suave y rápido. El subproceso de la aplicación JavaFX es un hilo diferente al Swing y AWT Distribuidor de Eventos Thread (EDT), por lo que se debe tener cuidado al incrustar código JavaFX en aplicaciones Swing.

- **Prism render thread**: Este thread se encarga de la representación por separado del distribuidor de eventos(EDT). Permite la trama(frame) N se represente mientras que la trama N +1 se está procesando. Esta capacidad de llevar a cabo el procesamiento simultáneo es una gran ventaja, especialmente en los sistemas modernos que tienen varios procesadores. El hilo Prism también puede tener múltiples hilos de rasterización que ayudan el trabajo fuera de la carga que hay que hacer en la representación.

- **Media thread**: Este se ejecuta en segundo plano y sincroniza los últimos frames a través del escenario gráfico utilizando el subproceso de la aplicación JavaFX.

Pulse(Pulsaciones)

Un pulso es un evento que indica al escenario gráfico JavaFX que es hora de sincronizar el estado de los elementos en el escenario gráfico con el hilo Prism. Un pulso es regulado a 60 frames por segundo (fps) como máximo y se dispara cada vez que las animaciones se ejecutan en el escenario gráfico. Incluso cuando la animación no se está ejecutando, el pulso se programa cuando se cambia algo en el escenario gráfico. Por ejemplo, si se cambia la posición de un botón, se programa un pulso.

Cuando se dispara un pulso, el estado de los elementos en el gráfico de la escena se sincroniza hacia abajo a la capa de renderizado. Un pulso permite a los desarrolladores de aplicaciones una forma de manejar los eventos de forma asincrónica. Esta importante función permite al sistema de proceso por lotes(batch) y ejecutar eventos en el pulso.

Maquetación y CSS también están ligados a eventos de pulso. Numerosos cambios en el escenario gráfico podrían llevar a múltiples maquetaciones o actualizaciones de CSS que podrían degradar seriamente el funcionamiento. El sistema efectúa automáticamente un CSS y fase de diseño una vez por pulso para evitar la degradación del rendimiento. Los desarrolladores de aplicaciones también pueden desencadenar de forma manual el diseño , según sea necesario para tomar medidas antes de un pulso.

El Windowing Toolkit Glass es responsable de ejecutar los eventos de pulso. Utiliza los temporizadores nativos de alta resolución para realizar la ejecución.

Imágenes y Multimedia(Media and Images)

JavaFX funcionalidad multimedia se encuentra disponible a través de las APIs javafx.scene.media. JavaFX es compatible con ambos medios visuales y de audio. Se proporciona soporte para MP3, AIFF, y WAV archivos de audio y archivos de vídeo FLV. JavaFX funcionalidad multimedia se proporciona como tres componentes separados: el objeto Media representa un archivo multimedia, el MediaPlayer reproduce un archivo multimedia, y una MediaView es un nodo que muestra los medios de comunicación.

El componente del motor de Media, que se muestra en verde en la figura 2-1, se ha diseñado con el rendimiento y la estabilidad en mente y proporciona un comportamiento coherente en todas las plataformas. Para obtener más información, lea la incorporación de los Media Assets.

Componente Web(Web Component)

El componente Web es un control de JavaFX UI basado en Webkit, que proporciona un visualizador Web y la funcionalidad completa de

navegación a través de su API. Este componente del motor Web, que se muestra en color naranja en la Figura 2-1, se basa en WebKit, que es un motor de código abierto navegador web que soporte HTML5, CSS, JavaScript, DOM, y SVG. Permite a los desarrolladores a implementar las siguientes características en sus aplicaciones Java:

- Renderizar contenido HTML de URL local o remoto
- Apoyar la historia y ofrecer Atrás y Adelante de navegación
- Actualizar el contenido
- Aplicar efectos al componente web
- Editar el contenido HTML
- Ejecutar comandos de JavaScript
- Manejar eventos

Este componente del navegador integrado se compone de las siguientes clases:

WebEngine proporciona capacidades básicas de la página web de navegación.

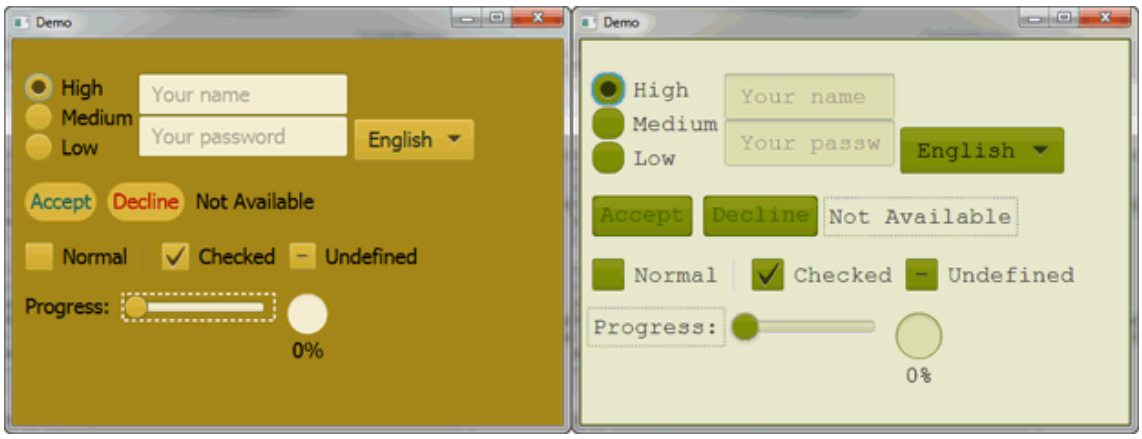
WebView encapsula un objeto WebEngine, incorpora el contenido HTML en la scene de una aplicación, y proporciona campos y métodos para aplicar efectos y transformaciones. Es una extensión de una clase Node.

Además, las llamadas de Java pueden ser controlados a través de JavaScript y viceversa para permitir a los desarrolladores hacer lo mejor de ambos entornos. Para información general más detallada del navegador integrado JavaFX, consulte la Adición de contenido HTML para JavaFX documento Aplicaciones.

Hojas de estilo en cascada CSS

Ofrece la posibilidad de aplicar un estilo personalizado a la interfaz de usuario de una aplicación JavaFX sin cambiar ningún de código fuente de la aplicación. CSS se puede aplicar a cualquier nodo en el gráfico de la escena JavaFX y se aplica a los nodos de forma asincrónica. Estilos CSS JavaFX también pueden ser fácilmente asignados a la escena en tiempo de ejecución, lo que permite la aparición de una aplicación para cambiar dinámicamente.

La figura 2-2 muestra la aplicación de dos estilos CSS diferentes para el mismo conjunto de controles de interfaz de usuario.



JavaFX CSS se basa en el W3C CSS versión 2.1 especificaciones, con algunas adiciones de trabajo actual de la versión 3. El soporte y las extensiones en JavaFX CSS se han diseñado para permitir que las hojas de estilo CSS para JavaFX pueden analizarse limpiamente por cualquier analizador sintáctico CSS, incluso uno que no admite extensiones de JavaFX. Esto permite que la mezcla de estilos CSS para JavaFX y para otros fines (por ejemplo, para las páginas HTML) en una única hoja de estilo. Todos los nombres de propiedades JavaFX están prefijados con una extensión de proveedor de "-fx-", incluidas las que pudieran parece compatible con el CSS HTML estándar, debido a que algunos valores de JavaFX tienen una semántica ligeramente diferente.

Para obtener información más detallada acerca de JavaFX CSS, vea Maquillando aplicaciones JavaFX con CSS.

Controles de intefaz de usuario(UI Controls)

Los controles de interfaz de usuario JavaFX están disponibles a través de la API de JavaFX, se construyen mediante el uso de nodos en el escenario gráfico(Scene Graph). Ellos pueden sacar el máximo provecho de las características de gran riqueza visual de la plataforma JavaFX y son portátiles a través de diferentes plataformas. JavaFX CSS permite una tematización y el maquillado de los controles de interfaz de usuario.

La Figura 2-3 muestra algunos de los controles de interfaz de usuario(UI) que se admiten actualmente. Estos controles se encuentran en el paquete javafx.scene.control.

Figure 2-3 JavaFX UI Controls Sample

Para obtener información más detallada acerca de todos los controles disponibles de JavaFX UI, consulte el documento Uso de controles de interfaz de usuario JavaFX y la documentación de la API para el paquete javafx.scene.control.

(Diseñadores, Esquematizadores, Gestores, Distribuidores)Layout

Contenedores de diseño o paneles pueden ser utilizados para prever disposiciones flexibles y dinámicas de los controles UI dentro de un escenario gráfico de una aplicación JavaFX. La API de JavaFX Layout incluye las siguientes clases de contenedor(container) que automatizan los modelos disposición(layouts) frecuentes:

- La clase BorderPane expone sus nodos de contenido en la parte superior, inferior, derecha, izquierda, centro o región.
- La clase HBox organiza sus nodos de contenido horizontalmente en una sola fila.
- La clase VBox organiza sus nodos de contenido verticalmente en una sola columna.

- La clase StackPane coloca sus nodos de contenido en una sola pila de atras hacia delante.

- La clase GridPane permite al desarrollador crear una cuadrícula flexible de filas y columnas en la que se diseña nodos de contenido.

- La clase FlowPane organiza sus nodos de contenido, ya sea en horizontal o vertical "flujo", envolviendo en el ancho especificado (para horizontal) o altura (vertical) de los límites.

- La clase TilePane coloca sus nodos de contenido en las celdas de diseño de tamaño uniforme o tejas

- La clase AnchorPane permite a los desarrolladores crear nodos de ancla en la parte superior, inferior, izquierda o centro del diseño.

Para lograr una estructura de disposición deseada, distintos contenedores pueden anidarse dentro de una aplicación JavaFX.

Para obtener más información sobre cómo trabajar con los layouts, vea el articulo Trabajo con layouts JavaFX. Para obtener más información sobre la API Layout JavaFX, consulte la documentación de la API para el paquete javafx.scene.layout.

Transformaciones 2-D y 3-D

Cada nodo del scene graph JavaFX se puede transformar en coordenadas el x-y utilizando las siguientes clases javafx.scene.transform:

- translate - Mover un nodo de un lugar a otro a lo largo de la x, y, z planos relativos a su posición inicial.

- scale - Cambiar el tamaño de un nodo que aparezca ya sea grande o más pequeño en las x, y, z planos, dependiendo del factor de escala.

- shear - Girar un eje de manera que el eje x , y eje y ya no están perpendiculares. Las coordenadas del nodo se desplazan por los multiplicadores especificados.

- rotate - Girar un nodo alrededor de un punto de pivote especificado de la escena.

- affine - Realizar un mapeo lineal de 2-D/3-D coordina a otras coordenadas 2-D/3-D conservando las propiedades y 'rectas' «paralelas» de las líneas. Esta clase se debe utilizar con Translate, Scale, Rotate o Shear transformará las clases en lugar de ser utilizado directamente.

Para obtener más información acerca de cómo trabajar con las transformaciones, vea Aplicando transaformaciones en JavaFX. Para obtener más información acerca de las clases de la API javafx.scene.transform, consulte la documentación de la API.

Efectos Visuales

El desarrollo de interfaces de cliente enriquecidas(rich intefaces) en el escenario gráfico(Scene Graph) JavaFX implica el uso de efectos visuales o de efectos para mejorar el aspecto de las aplicaciones JavaFX en tiempo real. Los efectos JavaFX son principalmente de imágenes de píxeles-based y, por lo tanto, ellos toman el conjunto de nodos que se encuentran en el escenario gráfico(Scene Graph), hacen que sea como una imagen, y aplicar los efectos establecidos para ella.

Algunos de los efectos visuales disponibles en JavaFX incluyen el uso de las siguientes clases:

- Drop Shadow - Renderiza una sombra de un contenido dado detrás del contenido al que se aplica el efecto.
- Reflection - Renderiza una versión reflejada del contenido a continuación el contenido real.
- Lighting - Simula una fuente de luz que brilla en un contenido dado y puede dar un objeto plano un aspecto más realista, tridimensional.

Para ver ejemplos de cómo utilizar algunos de los efectos visuales disponibles, consulte el documento Creación de Efectos Visuales. Para obtener más información acerca de todas las clases de efectos visuales disponibles, consulte la documentación de la API para el paquete javafx.scene.effect.

Primeros pasos con JavaFX Aplicaciones de ejemplo

Esta colección de aplicaciones de ejemplo está diseñado para ayudarle a empezar con las tareas de JavaFX comunes, incluyendo el trabajo con layouts, controles, hojas de estilo, FXML y efectos visuales.

- Hello World, Estilo JavaFX
- Diseño de formularios en JavaFX
- Diseño de lujo con CSS
- Diseño de interfaz de usuario con FXML
- Formas animados y efectos visuales

Hola Mundo al estilo JavaFX

La mejor manera de enseñarle lo que es crear y construir una aplicación JavaFX es con una aplicación "Hola Mundo". Un beneficio adicional de este tutorial es que le permite probar que su tecnología JavaFX se ha instalado correctamente.

La herramienta utilizada en este tutorial es NetBeans IDE 8. Antes de empezar, asegúrese de que la versión de NetBeans IDE que está usando admite JavaFX 8. Consulte la sección Sistemas, Configuración y Certificaciones de la página de Java SE 8 descargas para más detalles.

Construya la aplicación

1. En el menú **Archivo**, seleccione **Nuevo proyecto**.

2. En la categoría de aplicaciones **JavaFX**, elija **Aplicación JavaFX**. Haga clic en **Siguiente**.

3. Asigne un nombre al proyecto **HolaMundo** y haga clic en **Finalizar**.

   NetBeans abre el archivo HolaMundo.java y la rellena con el código de una aplicación básica de Hola Mundo, como se muestra en el ejemplo 3-1.

```
package holamundo;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HolaMundo extends Application {

    @Override
    public void start(Stage stage) {
        Button boton = new Button();
        boton.setText("Di 'Hola Mundo'");
        boton.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hola Mundo!");
            }
        });

        StackPane layout = new StackPane();
        layout.getChildren().add(boton);

 Scene escena = new Scene(layout, 300, 250);

        stage.setTitle("Hola Mundo!");
        stage.setScene(escena);
        stage.show();
    }
 public static void main(String[] args) {
        launch(args);
    }
}
```

Estas son las cosas importantes que debe saber acerca de la estructura básica de una aplicación JavaFX:

- La clase principal para una aplicación JavaFX amplía(extends) la clase javafx.application.Application. El método start() es el principal punto de entrada para todas las aplicaciones JavaFX.

- Una aplicación JavaFX define el contenedor(container) de interfaz de usuario(UI) por medio de una etapa(Stage) y una escena(Scene). La clase de JavaFX “Stage” es el contenedor JavaFX de nivel superior. La clase de JavaFX “Scene” es el contenedor de todo el contenido. Ejemplo 3-1 crea el escenario(Stage) y la escena(Scene) y hace que la escena se visualize en un tamaño de píxel dado.

- En JavaFX, el contenido de la escena se representa como un gráfico de la escena jerárquica de nodos. En este ejemplo, el nodo raíz es un objeto StackPane, que es un nodo de diseño(layout) de tamaño variable. Esto significa que el tamaño del nodo de la raíz escucha el tamaño de la escena y cambia cuando se cambia el tamaño de la etapa por un usuario.

- El nodo raíz contiene un nodo hijo, un control de botón con texto, además de un controlador de eventos para imprimir un mensaje cuando se pulsa el botón.

- El método main() no es necesario para las aplicaciones JavaFX cuando se crea el archivo JAR para la aplicación con la herramienta de JavaFX Packager que incrusta el Lanzador JavaFX(JavaFX Launcher) en el archivo JAR. Sin embargo, es útil incluir el método main() para que pueda ejecutar archivos JAR que se crearon sin el Lanzador JavaFX (JavaFX Launcher), como cuando se utiliza un IDE en el que las herramientas de JavaFX no están totalmente integradas. Además, las aplicaciones Swing que incrustan código JavaFX requieren el método main().

La Figura 3-1 muestra el escenario gráfico(Scene Graph) para la aplicación Hello World. Para obtener más información sobre los escenarios gráficos consulte Uso de Scene Graph de JavaFX.

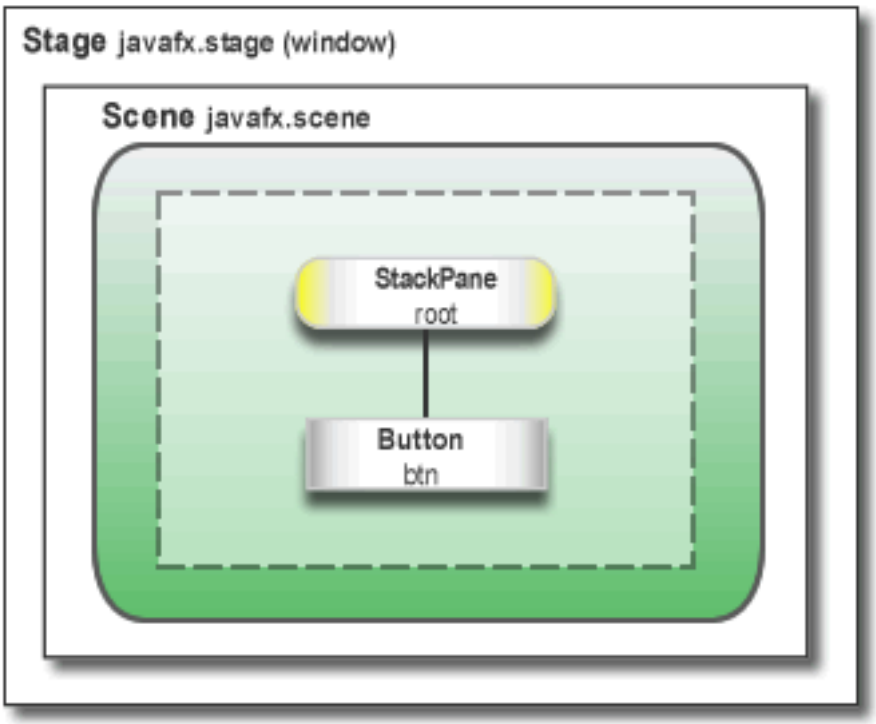


Figure 3-1 Hello World Scene Graph

Ejecutar la aplicación

En la ventana Proyectos, haga clic en el nodo del proyecto **HolaMundo** y seleccione **Ejecutar**.

Haga clic en el botón Dí Hola Mundo.

Compruebe que el texto "Hola Mundo!" Se imprime en la ventana de salida de NetBeans.
La figura 3-2 muestra la aplicación Hola Mundo, al estilo JavaFX.

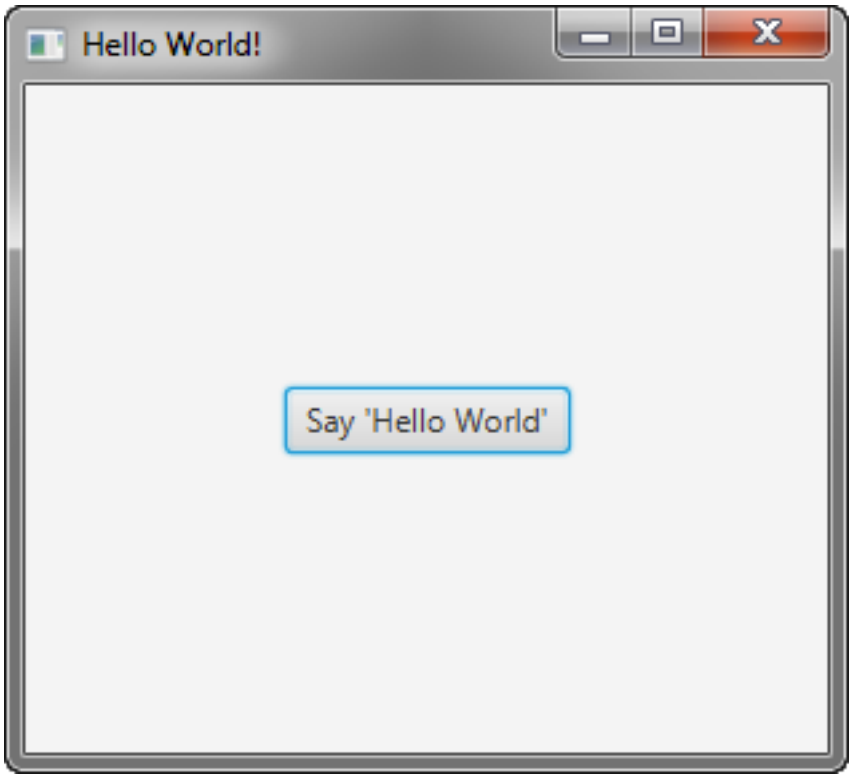


Figura 3-2 Hola Mundo, al estilo de JavaFX.

Donde ir a continuación:

Esto concluye el tutorial básico Hello World, pero sigue leyendo para más lecciones sobre el desarrollo de aplicaciones JavaFX:

- Crear un formulario en JavaFX, enseña los fundamentos de diseño de la pantalla, cómo agregar controles a un diseño, y cómo crear eventos de entrada.
- Formas de lujo(Fancy forms) con JavaFX y CSS proporciona trucos de estilo simple para la mejora de su aplicación, como agregar una imagen de fondo y los botones de estilo y texto.
- Usando FXML para crear una interfaz de usuario muestra un método alternativo para crear la interfaz de usuario de inicio de sesión. FXML es un lenguaje basado en XML que proporciona la estructura para la construcción de una interfaz de usuario independiente de la lógica de la aplicación de su código.
- Animación y Efectos Visuales en JavaFX muestra cómo llevar a cabo una aplicación a la vida mediante la adición de animación, línea de tiempo y efectos de mezcla.

Creando un formulario en javaFX

La creación de un formulario es una actividad común en el desarrollo de una aplicación. Este tutorial te enseña los fundamentos de los gestores de diseño(layout) en la pantalla, cómo agregar controles a un panel de diseño(layout), y cómo crear eventos de entrada.

En este tutorial, utilizará JavaFX para construir el formulario de acceso se muestra en la Figura 4-1.

Figura 4-1 Formulario de acceso

La herramienta utilizada en este tutorial de introducción es NetBeans IDE. Antes de empezar, asegúrese de que la versión de NetBeans IDE que está usando admite JavaFX 8. Vea la página de Descargas y Configuraciones del sistema, de la página de Java SE Descargas para más detalles.

Creando el Proyecto

Tu primer tarea es crear un proyecto en NetBeans IDE y llamarlo Login:

1. En el menu **Archivo**, seleccione **Nuevo Proyecto**.
2. En la categoría de aplicaciones javaFX, elija **Aplicación JavaFX**. Haga clic en **siguiente**.
3. El nombre del proyecto será: **Login** y haga clic en **Finalizar**.

Al crear un proyecto JavaFX NetBeans IDE proporciona una aplicación Hola Mundo como un punto de partida, que ya ha visto si ha seguido el tutorial Hola Mundo.

4. Retire el método start() que NetBeans IDE genera y reemplacelo con el código en el ejemplo 4-1.

Ejemplo 4-1 Etapa de Aplicación

```
@Override
public void start(Stage primaryStage)
{
    primaryStage.setTitle("JavaFX Welcome");
    primaryStage.show();
}
```

Consejo: Después de agregar el código de ejemplo en un proyecto de NetBeans, pulse Ctrl (o Cmd) + Shift + I para importar los paquetes necesarios. Cuando hay una selección de declaraciones de importación, elija el que comienza con JavaFX.

Crear un GridPane Layout

Para el formulario de acceso que vamos a crear, utilizaremos un diseño(layout) "GridPane" porque te permite crear una cuadrícula flexible de filas y columnas en la que se pueden colocar controles. Usted puede colocar los controles en cualquier celda de la cuadrícula, y usted puede hacer que los controles abarquen las celdas que según sean necesarias.

El código para crear el diseño(layout) es GridPane en el ejemplo 4-2. Agregue el código delante de la línea primaryStage.show();

Ejemplo 4-2 GridPane con (huecos)Gap y propiedades de relleno(padding)

```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));

Scene scene = new Scene(grid, 300, 275);
primaryStage.setScene(scene);
```

Ejemplo 4-2 crea un objeto GridPane y lo asigna a la variable llamada "grid". La propiedad de alineación(`.setAlignment()`) cambia la posición por defecto de la rejilla(`grid`) de la parte superior izquierda de la escena hacía el centro. Las propiedades Gap gestionan el espaciado entre las filas y las columnas, mientras que la propiedad de relleno(`padding`) gestiona el espacio alrededor de los bordes del panel Cuadrícula(`grid pane`). Las inserciones están en el orden de arriba, derecha, abajo e izquierda. En este ejemplo, hay 25 píxeles de relleno(`padding`) en cada lado.

La escena se crea con el panel de rejilla(`grid`) como el nodo raíz(`root`), que es una práctica común cuando se trabaja con contenedores(`containers`) de diseño(`layout`).

Por lo tanto, ya que la ventana cambia de tamaño, los nodos dentro del panel de cuadrícula cambian de tamaño en función de sus limitaciones(constraints) de diseño(layout). En este ejemplo, el panel de rejilla(grid) permanece en el centro cuando se hace grande o bien cuando reduce la ventana. Las propiedades de relleno(padding) asegúrese de que no es un relleno alrededor del panel de cuadrícula al hacer la ventana más pequeña.

Este código establece el ancho y la altura de la escena al 300px por 275px. Si no ajusta las dimensiones de la escena, los valores predeterminados de la escena tomarán el mínimo necesario para mostrar su contenido.

Añadir texto(Text), etiquetas(Label) y campos de texto(TextField)

En cuanto a la Figura 4-1, se puede ver que el formulario requiere el título de "Bienvenido" y los campos de texto(TextField) y campo de contraseña(PasswordField) para obtener la información por parte del usuario. El código para la creación de estos controles esta en el Ejemplo 4-3. Agregue este código después de la línea que establece la propiedad de relleno(padding) de cuadrícula(grid).

```
Text scenetitle = new Text("Welcome");
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
grid.add(scenetitle, 0, 0, 2, 1);

Label userName = new Label("User Name:");
grid.add(userName, 0, 1);

TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);

Label pw = new Label("Password:");
grid.add(pw, 0, 2);

PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);
```

La primera línea crea un objeto de texto(Text) que no se puede editar, establece el texto de “Bienvenida”, y lo asigna a una variable llamada “scenetitle”. En la siguiente línea se utiliza el método setFont() para establecer la familia de fuentes(FontFamily), el peso(weight) y el tamaño(size) de la variable scenetitle. Usando un estilo en línea es apropiado cuando el estilo se une a una variable, pero una mejor técnica para estilizar los elementos de su interfaz de usuario(UI) es mediante el uso de una hoja de estilo(CSS). En el siguiente tutorial, Formulario con JavaFX y CSS, reemplazará el estilo en línea con una hoja de estilos(Cascading Style Sheet).

El método grid.add(), añade la variable scenetitle a la cuadricula(grid) de distribución(layout). La numeración de las columnas y las filas de la cuadrícula comienza en cero, y scenetitle se añade en la columna 0 y en la fila 0. Los dos últimos argumentos del método grid.add() establecen el espacio(span) de la columna a 2 y el lapso(span) de la fila 1.

Las líneas siguientes crean un objeto Label con el texto Nombre de Usuario(`User Name:`) en la columna 0 y en la fila 1, y un objeto de campo de texto(`TextField`) que se puede editar. El campo de texto se añade al panel de cuadrícula(`GridPane`) en la columna 1, fila 1. Un campo de contraseña(`PasswordField`) y las demas etiquetas se crean y se añaden al panel de caudricula(`GridPane`) de una manera similar.

Cuando se trabaja con un panel de cuadrícula(`GridPane`), puede mostrar las líneas de cuadrícula, que es útil para propósitos de depuración. En este caso, puede agregar `grid.setGridLinesVisible(true)` después de la línea que agrega el campo de contraseña. Entonces, cuando se ejecuta la aplicación, verá las líneas de las columnas y filas de la cuadrícula, así como las propiedades de la brecha(`gap`), como se muestra en la Figura 4-2.

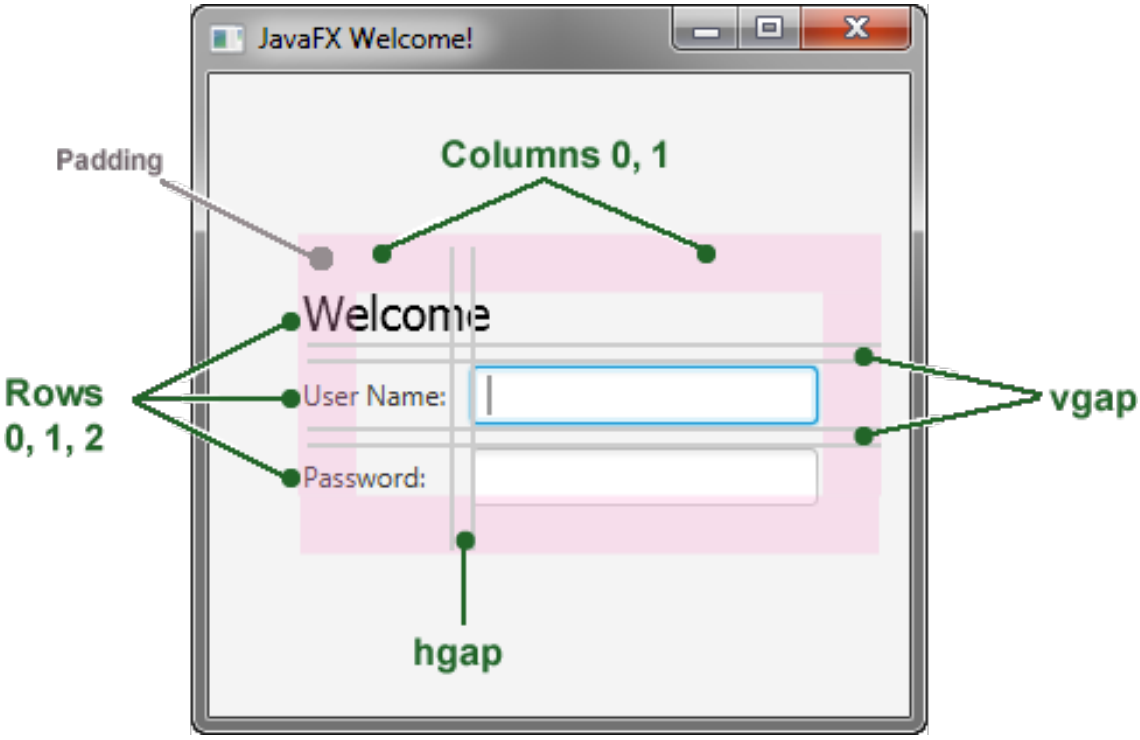


Figura 4-2 Formulario de Login en modo cuadricula

Agregar un botón y texto

Los dos últimos controles necesarios para la aplicación son un control Botón(`Button`) para la presentación de los datos y un control de texto(`Text`) para mostrar un mensaje cuando el usuario pulsa el botón.

Primero, cree el botón y colóquelo en la parte inferior derecha, que es una posición común para los botones que realizan una acción que afecta a todo el formulario. Ejemplo 4-4. Agregue este código antes del código para la escena(`Scene`).

Ejemplo 4-4 Botón(Button)

```
Button btn = new Button("Sign in");
HBox hbBtn = new HBox(10);
```

```
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 4);
```

La primera línea crea un botón llamado "**btn**" con la etiqueta Entrar(**Sign in**), y la segunda línea crea un panel de diseño(**layout**) **HBox** llamado "**hbBtn**" con una separación de 10 píxeles para los lados. El panel de **HBox** establece una alineación para el botón que es diferente de la alineación se aplica a los otros controles en el panel de rejilla(**GridPane**). La propiedad de alineación(`.setAlignment()`) tiene un valor de **Pos.BOTTOM_RIGHT**, que posiciona a un nodo en la parte inferior del espacio verticalmente y en el borde derecho del espacio horizontal. Se añade el botón como un hijo(**Child**) del panel **HBox** y el panel **HBox** se añade a la red(**grid**) en la columna 1, fila 4.

Ahora, agregue un control de texto(`Text`) para mostrar el mensaje, como se muestra en el ejemplo 4-5. Agregue este código antes del código para la escena(`Scene`).

Ejemplo 4-5 Text

```
final Text actiontarget = new Text();
grid.add(actiontarget, 1, 6);
```

Figura 4-3 muestra el formulario ahora. Usted no verá el mensaje de texto hasta que lo trabaje a través de la siguiente sección del tutorial, Agregue código para controlar un evento.

Figura 4-3 Formulario Login con Botón

Agregue código para controlar un evento

Por último, para hacer que el botón de visualicé el mensaje de texto cuando el usuario lo presiona tendremos que añadir el código en el Ejemplo 4-6 antes que el código de la escena(Scene).

Ejemplo 4-6 Eventos del botón

```
btn.setOnAction(new EventHandler<ActionEvent>()
{
    @Override
    public void handle(ActionEvent e)
    {
        actiontarget.setFill(Color.FIREBRICK);
        actiontarget.setText("Sign in button pressed");
    }
});
```

El método setOnAction() se utiliza para registrar un controlador de eventos(EventHandler) que establece el objeto actiontarget entrar(Sign in) en el botón presionado cuando el usuario pulsa el botón. El color del objeto actiontarget(actiontarget se refiere al objeto objetivo, es decir en este caso será el Texto de aviso) se establece en **"Firebrick"** rojo.

Ejecutando la aplicación

Haga clic derecho en el archivo **Login.java** del proyecto en la ventana Proyectos, seleccione **Ejecutar** y a continuación haga clic en el botón Entrar(Sign in). La Figura 4-4 muestra los resultados. Si se encuentra con problemas, a continuación eché un vistazo a el código en el archivo Login.java que se incluye en el archivo Login.zip descargable.

Ejemplo 4-4 Formulario Login final

Donde ir ahora?

Con esto concluye el tutorial básico del Formulario, pero puede seguir leyendo los siguientes tutoriales sobre el desarrollo de aplicaciones JavaFX.

- El Formulario de lujo con JavaFX y CSS proporciona consejos acerca de cómo agregar una imagen de fondo y cambiar radicalmente el estilo del texto, la etiqueta y el botón en el formulario de acceso.

- Usando FXML para crear una interfaz de usuario muestra un método alternativo para crear la interfaz de usuario de inicio de sesión. FXML es un lenguaje basado en XML que proporciona la estructura para la construcción de una interfaz de usuario independientemente de la lógica de la aplicación de su código.

- Trabajar con layouts de JavaFX explica los paneles de JavaFX incorporados(Built-in), consejos y trucos para el uso de ellos.

También probar los ejemplos de JavaFX, que se puede descargar desde la sección de la página de Descargas Demos y Muestras de Java SE JDK en http://www.oracle.com/technetwork/java/javase/downloads/. El ejemplo Ensemble contiene ejemplos de los diseños(Layouts) y su código fuente.

Formulario con CSS en javaFX

Este tutorial es acerca de hacer su aplicación JavaFX más atractiva, añadiendo una hoja de estilos en cascada(Cascading Style Sheet). Desarrolla un diseño, crea un archivo css, y aplique los nuevos estilos.

En este tutorial, tomaremos el formulario de acceso que utiliza estilos por defecto para las etiquetas, los botones, y el color de fondo, con algunas modificaciones de estilo CSS simples, la convertirá en una aplicación estilizada, como se muestra en la Figura 5-1.

Figura 5-1 Formulario Login sin Css y con Css

La herramienta utilizada en este tutorial de introducción es NetBeans IDE. Antes de empezar asegúrese de que la versión de NetBeans IDE que está usando admite JavaFX 8. Vea la página de Configuraciones del sistema en la página de Java SE Descargas para más detalles.

Creando el Proyecto

Si has seguido la guía **Primeros pasos con JavaFX Aplicaciones,** entonces usted ya ha creado el proyecto necesario para ingresar a este tutorial. Si no, descargue el proyecto Login haciendo clic derecho en Login.zip y guardelo en su sistema de archivos. Extraiga los archivos del archivo zip y luego abra el proyecto en NetBeans IDE.

Creand el archivo CSS

Su primera tarea es crear un nuevo archivo CSS y guárdalo en el mismo directorio que la clase principal de la aplicación. Después de eso, usted debe hacer que la aplicación JavaFX tome de la hoja de estilos en cascada que acaba de agregar.

1. En la ventana de NetBeans IDE Proyectos, expanda el nodo del Proyecto **Login** y luego el nodo de **directorio de paquetes(source packages)** fuente.

2. Haga clic derecho debajo de **directorio de paquetes(source packages)** y en el paquete actual de trabajo, seleccione **Nuevo** y a continuación **Otros**

3. En el cuadro de diálogo Nuevo archivo, seleccione **Otro**, de **Cascading Style Sheet**, y haga clic en **Siguiente**.

4. Coloque el nombre de **Login** para el campo de texto en Nombre de archivo y asegurese de que el valor del campo de texto de la carpeta este en el mismo lugar que el del código fuente.

5. Clic en **Finalizar**.

6. En el archivo Login.java, se inicializa la variable de hojas de estilo de la "clase escena(Scene)" para que apunte a la hoja de estilos en cascada mediante la inclusión de la línea de código que se muestra a continuación en negrita para que aparezca como se muestra en el ejemplo 5-1.

   Ejemplo 5-1 Inizialización de la variable

   ```
   Scene scene = new Scene(grid, 300, 275);
   primaryStage.setScene(scene);
   scene.getStylesheets().add(
   Login.class.getResource("Login.css").toExternalForm();
   primaryStage.show();
   ```

   Este código busca la hoja de estilos en el directorio de src\login en el proyecto de NetBeans.

Agregando una imágen de fondo

Una imágen de fondo ayuda a que su formulario sea más atractivo. Para este tutorial, agregará un fondo gris con una textura similar al lino.

En primer lugar, descargue la imágen de fondo, haciendo clic derecho en la imagen background.jpg y guardelo en la carpeta src\login en el proyecto Login dentro de NetBeans.

Ahora, agregue el siguiente código, la propiedad: background-image en el archivo CSS. Recuerde que la la ruta(path) es relativa a la hoja de estilos. Así que en el código del ejemplo 5-2, la imagen background.jpg está en el mismo directorio que el archivo Login.css.

Ejemplo 5-2 Fondo gris con textura similar a lino.

Aplicando estilo a las etiquetas(Labels)

Los siguientes controles para mejorar son las etiquetas. Usarás .label un estilo de clase, Estilo de etiqueta(Label), que significa que los estilos afectarán a todas las etiquetas en el formulario. El código es en el ejemplo 5-3.

Figura 5-3 Etiquetas con mayor grosor y efecto de sobre

Estilo de Textos(Text)

Ahora, crear algunos efectos especiales en los dos objetos de texto en el formulario: “scenetitle”, que incluye el texto de bienvenida, y “actiontarget”, que es el texto que se devuelve cuando el usuario pulsa el botón Entrar(Sign in). Puede aplicar diferentes estilos a los objetos de texto utilizados en este tipo de maneras diversas.

1. En el archivo Login.java, elimine las siguientes líneas de código que definen los estilos en línea(inline styles) actualmente establecidos para los objetos de texto:

   ```
   scenetitle.setFont(Font.font(”Tahoma”, FontWeight.NORMAL, 20));
   actiontarget.setFill(Color.FIREBRICK);
   ```

   Al cambiar a CSS sobre los estilos en línea, se separa el diseño del contenido. Este enfoque hace que sea más fácil para un diseñador para tener un control sobre el estilo sin tener que modificar contenido.

2. Crea un identificador(ID) para cada nodo de texto utilizando el método setId() de la clase **Node**: Agregue las siguientes líneas en negrita para que aparezcan como se muestra en el ejemplo 5-4.

Ejemplo 5-4 Crea un ID para los nodos de Texto

```
..
Text scenetitle = new Text("Welcome");
scenetitle.setId("welcome-text");
..
..
grid.add(actiontarget, 1, 6);
actiontarget.setId("actiontarget");
..
```

3. En el archivo Login.css, define las propiedades de estilo para los ID's "welcome-text" y "actiontarget". Para poder usar los estilos por nombre de ID, se utiliza el simbolo numeral "#" como se muestra en la en el Ejemplo 5-5.

Ejemplo 5-5 Efectos en Texto

```
#welcome-text
{
   -fx-font-size: 32px;
   -fx-font-family: "Arial Black";
   -fx-fill: #818181;
   -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6, 0.0 , 0 , 2 );
}
#actiontarget
{
  -fx-fill: FIREBRICK;
  -fx-font-weight: bold;
  -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

4. El tamaño del texto de "Welcome" se aumenta a 32 puntos y la fuente se cambia a Arial Negro. El color de relleno(fill) de texto se establece en un color gris oscuro (#818181) y se aplica un efecto de sombra interior, creando un efecto de relieve. Puede aplicar una sombra interior en cualquier color, cambiando el color de relleno(fill) de texto para hacer una versión más oscura del fondo. Vea la sección sobre los efectos en la Guía de referencia CSS JavaFX para obtener detalles sobre los parámetros de la propiedad sombra interior(inner shadow property).

   La definición de estilo para **actiontarget** es similar a lo que has visto hace poco.

   La Figura 5-4 muestra los cambios de fuente y efectos de sombra en los dos objetos de texto.

Figura 5-4 Texto con efectos de sombra

Aplicando estilo al Botón

El siguiente paso es el estilo del botón, por lo que es el cambio de estilo cuando el usuario pasa el ratón sobre él. Este cambio dará a los usuarios una indicación de que el botón es interactivo, una práctica estándar de diseño.

En primer lugar, crear el estilo para el estado inicial del botón con el código en el ejemplo 5-6. Este código utiliza el selector de clase estilo de botón "**.button**", de manera que si se agrega un botón al formulario en una fecha posterior, a continuación, el nuevo botón también utilizar este estilo.

Ejemplo 5-6 efecto Drop Shadow para el Botón(Button)

```
.button
{
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}
```

Ahora, creareamos un aspecto ligeramente diferente para cuando el usuario pasa el ratón por encima del botón. Esto se hace con la pseudo-clase **"hover"**. Una pseudo-clase incluye el selector para la clase y el nombre para el estado separado por dos puntos (:), como se muestra en el ejemplo 5-7.

Ejemplo 5-7 Botón con estilos Hover

```
.button:hover
{
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}
```

La figura 5-5 muestra el estado inicial y hover del botón con su nuevo fondo azul-gris y blanco negrita.

Figura 5-5 Estado incial y Hover del botón

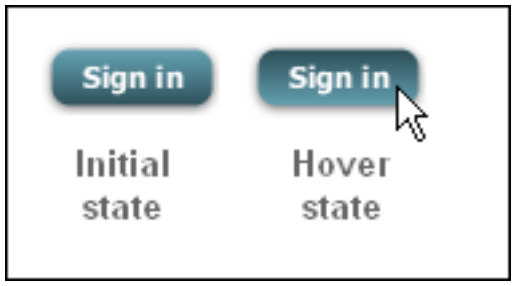


Figura 5-6 muestra la aplicación final

Figura 5-6 Aplicación con estilos finalizada

Donde ir ahora?

Aqui estan algunas cosas que deberías intentar acontinuación:

Vea lo que puede crear usando CSS. Algunos documentos que le pueden ayudar a que usted son Decorando aplicaciones JavaFX con CSS, Styling Gráficos con CSS y la Guía de referencia CSS JavaFX. El Skinning con CSS y la sección Analizador CSS de la Guía del usuario de JavaFX Scene Builder también proporciona información sobre cómo se puede utilizar la herramienta de escena JavaFX Builder para la carcasa de su diseño JavaFX FXML.

Vea Estilos en Botones FX con CSS para ver ejemplos de cómo crear estilos comunes del botón usando CSS.

Usando FXML para crear Interfaces de Usuario

Este tutorial muestra los beneficios del uso de JavaFX FXML, que es un lenguaje basado en XML que proporciona la estructura para la construcción de una interfaz de usuario independiente de la lógica de la aplicación de su código.

Si usted comenzó este documento desde el principio, entonces usted ha visto cómo crear una aplicación de inicio de sesión utilizando sólo JavaFX. Aquí, se utiliza FXML para crear la misma interfaz de usuario de inicio de sesión, que separa el diseño de la aplicación de la lógica de la aplicación, con lo que el código sea más fácil de mantener. La interfaz de usuario de inicio de sesión se genera en este tutorial se muestra en la Figura 6-1.

Figura 6-1 Interfaz de usuario de Login

Este tutorial utiliza NetBeans IDE. Asegúrese de que la versión de NetBeans IDE que está usando admite JavaFX 8. Consulte la sección de Descargas en la página de Java SE Descargas para más detalles.

Configurando el proyecto

Tu primer tarea será configurar un Proyecto **Aplicación JavaFX FXML** en el IDE NetBeans:

1. En el menu **Archivo**, elija **Nuevo Proyecto**
2. En la categória de aplicación **JavaFX**, elija **Aplicación JavaFX FXML**. Luego clic en **Siguiente**.
3. Coloque el nombre al proyecto **FXMLExample** y clic en **Finalizar**.

    - FXMLExample.java. Este archivo se encarga del código Java estándar requerido para una aplicación FXML.

- FXMLDocument.fxml. Este es el archivo de origen FXML en el que se define la interfaz de usuario.

- FXMLDocumentController.java. Este es el archivo de controlador para manejar la entrada del ratón y el teclado.

4. Renombre el archivo FXMLDocumentController.java a FXMLExampleController.java para que el nombre sea mas significativo para esta aplicación.

    a. En la ventana Proyectos, haga clic con el botón FXMLDocumentController.java y seleccione Refactor luego Cambiar nombre.

    b. Introduzca FXMLExampleController, y haga clic en Refactorizar.
5. Renombre FXMLDocument.fxml a fxml_example.fxml.

    a. Haga clic derecho y seleccione Cambiar nombre FXMLDocument.fxml.

    b. Introduzca fxml_example y haga clic en Aceptar.

Cargue el archivo de origen FXML

El primer archivo a editar es el archivo FXMLExample.java. Este archivo incluye el código para la creación de la clase principal de la aplicación y para definir el escenario y escena. Más específico FXML, el archivo utiliza la clase FXMLLoader, que es responsable de cargar el archivo de origen FXML y devolver el gráfico de objetos resultante.

Realice los cambios que aparecen en negrita en el ejemplo 6-1.

Ejemplo 6-1 FXMLExample.java

```
@Override
public void start(Stage stage) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("fxml_example.fxml"));
    Scene scene = new Scene(root, 300, 275);
    stage.setTitle("FXML Welcome");
    stage.setScene(scene);
    stage.show();
}
```

Una buena práctica consiste en establecer la altura y la anchura de la escena cuando la crea, en este caso 300 por 275, de lo contrario los valores predeterminados de escena al mínimo necesario para mostrar su contenido.

Modificar las Declaraciones de Importación

A continuación, edite el archivo fxml_example.fxml. Este archivo especifica la interfaz de usuario que se muestra cuando se inicia la aplicación. La primera tarea consiste en modificar las declaraciones de importación por lo que su código es el Ejemplo 6-2.

Declaraciones Declaración XML Ejemplo 6-2 e Importar

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.net.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>
```

Al igual que en Java, nombres de clases pueden ser completo (incluyendo el nombre del paquete), o pueden ser importadas mediante la sentencia import, como se muestra en el ejemplo 6-2. Si lo prefiere, puede utilizar las declaraciones de importación específicas que se refieren a clases.

Crear un GridPane Layout

La aplicación Hello World generado por NetBeans utiliza un diseño AnchorPane. Para el formulario de acceso, se utilizará un diseño GridPane porque te permite crear una cuadrícula flexible de filas y columnas en la que se distribuyen controles.

Retire la disposición AnchorPane y sus hijos y reemplacelo con el layout GridPane en el Ejemplo 6-3.

Ejemplo 6-3 GridPane Layout

```
<GridPane fx:controller="fxmlexample.FXMLExampleController"
    xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
<padding>
    <Insets top="25" right="25" bottom="10" left="25"/>
</padding>

</GridPane>
```

En esta aplicación, el diseño GridPane es el elemento raíz del documento FXML y como tal tiene dos atributos. La fx:controller Se requiere atributo controlador cuando especifique manejadores de eventos basados en controladores en su margen de beneficio. Los atributo xmlns:fx siempre se requiere y especifica el espacio de nombres fx.

El resto del código controla la alineación y el espacio del panel de rejilla. La propiedad de alineación cambia la posición por defecto de la rejilla de la parte superior izquierda de la escena al centro. Las propiedades Gap gestionar el espaciado entre las filas y las columnas,

mientras que la propiedad de relleno gestiona el espacio alrededor de los bordes del panel Cuadrícula.

Como se cambia el tamaño de la ventana, los nodos dentro del panel de cuadrícula cambian de tamaño en función de sus limitaciones de diseño. En este ejemplo, la red permanece en el centro cuando se hace grande o reduce la ventana. Las propiedades de relleno(padding) asegúrese de que no es un relleno alrededor de la red cuando usted hace la ventana más pequeña.

Agregar campos de texto y Contraseña

Mirando hacia atrás en la Figura 6-1, se puede ver que la forma de inicio de sesión requiere el título de "Welcome" y los campos de texto y la contraseña para la captura de información por parte del usuario. El código en el Ejemplo 6-4 es parte del layout GridPane y debe ser colocado antes de la instrucción </GridPane>.

Controles Ejemplo 6-4 de texto, etiquetas, campos TextField y contraseña.

```
<Text text="Welcome" GridPane.columnIndex="0" GridPane.rowIndex="0"
GridPane.columnSpan="2"/>

<Label text="User Name:" GridPane.columnIndex="0" GridPane.rowIndex="1"/>

<TextField GridPane.columnIndex="1" GridPane.rowIndex="1"/>

<Label text="Password:" GridPane.columnIndex="0" GridPane.rowIndex="2"/>

<PasswordField fx:id="passwordField" GridPane.columnIndex="1" GridPane.rowIndex="2"/>
```

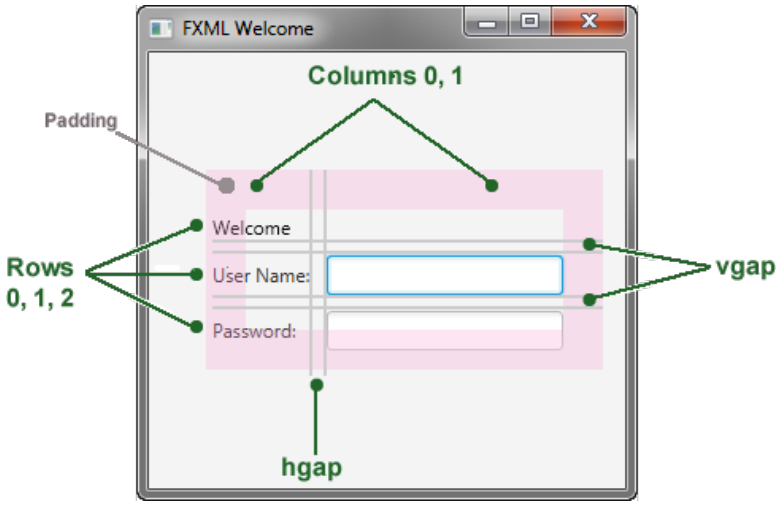
La primera línea crea un objeto de texto y establece el valor de texto “Welcome”. Los atributos GridPane.columnIndex y GridPane.rowIndex corresponden a la colocación del control de texto en la cuadrícula. La numeración de filas y columnas de la cuadrícula comienza en cero y la ubicación del control de texto se establece en (0,0), lo que significa que está en la primera columna de la primera fila. El atributo GridPane.columnSpan se establece en 2, por lo que la duración del título “Welcome” toma dos columnas de la cuadrícula. Usted necesitará esta anchura adicional más adelante en el tutorial al agregar una hoja de estilo para aumentar el tamaño de fuente del texto a 32 puntos.

Las líneas siguientes crean un objeto Label con texto User Name: en la columna 0, fila 1 y un objeto TextField a la derecha del mismo, en la columna 1, fila 1 . Otra etiqueta y el objeto campo Password: se crean y se añaden a la red de una manera similar.

Cuando se trabaja con un diseño de cuadrícula(GridPane), puede mostrar las líneas de cuadrícula, que es útil para propósitos de depuración. En este caso, establezca la propiedad gridLinesVisible como true mediante la adición de la declaración <gridLinesVisible>true</gridLinesVisible> justo después del la declaración <padding></padding>. Entonces, cuando se ejecuta la aplicación, verá las líneas de las columnas y filas de la cuadrícula,

así como las propiedades de la brecha(gap), como se muestra en la Figura 6-2.

Figura 6-2 Formulario Login con GridPane



Agregar un botón y texto

Los dos últimos controles necesarios para la aplicación son un control Button para la presentación de los datos y un control de texto para mostrar un mensaje cuando el usuario pulsa el botón. El código es en el Ejemplo 6-5. Añadir este código antes de </GridPane>.

Ejemplo 6-5 HBox, Button, y texto

```
<HBox
spacing="10" alignment="bottom_right" GridPane.columnIndex="1" GridPane.rowIndex="4" >
    <Button text="Sign In" onAction="#handleSubmitButtonAction"/>
</HBox>

<Text fx:id="actiontarget" GridPane.columnIndex="0" GridPane.columnSpan="2"
GridPane.halignment="RIGHT" GridPane.rowIndex="6"/>
```

Se necesita un panel de HBox para establecer una alineación para el botón que es diferente de la alineación predeterminada aplicada a los otros controles en el diseño GridPane. La propiedad de alineación se establece en BOTTOM_RIGHT, que posiciona a un nodo en la parte inferior del espacio verticalmente y en el borde derecho del espacio horizontal. El panel HBox se añade a la red en la columna 1, fila 4.

El panel HBox tiene un hijo, un botón con la propiedad de texto se establece en "Sign in" y una propiedad de conjunto `onAction` a `handleSubmitButtonAction()`. Mientras FXML es una manera conveniente para definir la estructura de la interfaz de usuario de una aplicación que no proporciona una manera de implementar el comportamiento de una aplicación. Implementa el comportamiento del método `handleSubmitButtonAction()` en el código de Java en la siguiente sección de este tutorial, agregue código para controlar un evento.

Asignación de un fx:id el valor de un elemento, como se muestra en el código para el Control del texto, se crea una variable en el espacio de nombres del documento, que se puede hacer referencia a partir en otra parte del código. Aunque no es necesario, la definición de un campo de control ayuda a aclarar cómo se relacionan el controlador y el marcado.

Agregar código para controlar un evento

Ahora haga el control de texto mostrará un mensaje cuando el usuario pulsa el botón. Esto se hace en el archivo FXMLExampleController.java. Elimine el código que NetBeans IDE genera y reemplazarlo con el código en el ejemplo 6-6.

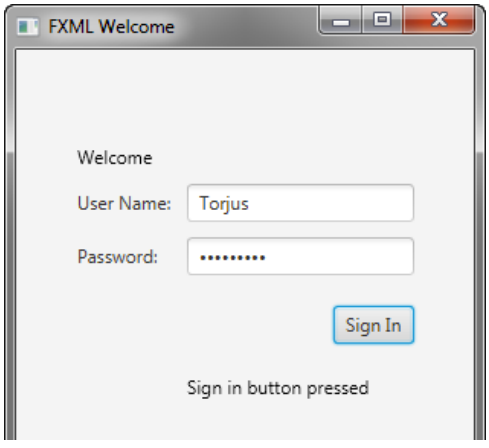
Ejemplo 6-6 FXMLExampleController.java

```
package fxmlexample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.text.Text;

public class FXMLExampleController
{
    @FXML
    private Text actiontarget;

    @FXML
    protected void handleSubmitButtonAction(ActionEvent event)
    {
        actiontarget.setText("Sign in button pressed");
    }
}
```

La anotación @FXML se utiliza para etiquetar campos miembros del controlador no públicos y los métodos de controlador para el uso de FXML. El método `handleSubmtButtonAction()` establece la variable `actiontarget` `Sign in` el botón presionado cuando el usuario pulsa el botón.

Puede ejecutar la aplicación ahora para ver la interfaz de usuario completa. La Figura 6-3 muestra los resultados cuando se escribe texto en los dos campos y haga clic en el botón `Sign in`. Si usted tiene algún problema, entonces usted puede comparar el código con el ejemplo FXMLLogin.

Figura 6-3 FXML Login Window



Use un lenguaje de Script para gestionar eventos

Como una alternativa al uso de código Java para crear un controlador de eventos, puede crear el controlador con cualquier lenguaje que proporciona un motor de scripting compatible con JSR 223. Algunos lenguajes incluidos son: JavaScript, Groovy, Jython, y Clojure.

Opcionalmente, puede probar a usar JavaScript ahora.

1. En el archivo fxml_example.fxml, agregue la declaración de JavaScript después de la declaración doctype XML.

   ```
   <?language javascript?>
   ```

2. En el marckup del botón, cambie el nombre de la función por el siguiente:

   ```
   onAction="handleSubmitButtonAction(event);"
   ```

3. Retire el atributo fx:controller marckup de GridPane y agregue la función de JavaScript en una etiqueta <script> directamente debajo de ella, como se muestra en el ejemplo 6-7.

   Ejemplo 6-7 JavaScript en FXML

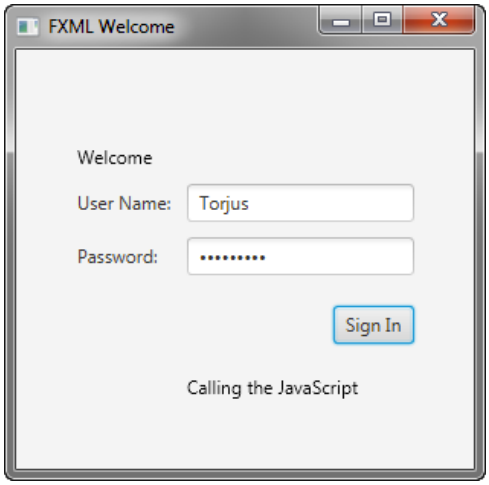
   ```
   <GridPane
   xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">

   <fx:script>
       function handleSubmitButtonAction()
       {
           actiontarget.setText("Calling the JavaScript");
       }
   </fx:script>
   ```

   Alternativamente, usted puede poner las funciones de JavaScript en un archivo externo (como por ejemplo: fxml_example.js) e incluir el script como el siguiente:

   ```
   <fx:script source="fxml_example.js"/>
   ```

   El resultado es en la Figura 6-4.

Figura 6-4 Inicio de sesión de aplicación mediante JavaScript



Si usted está considerando el uso de un lenguaje de script con FXML, tenga en cuenta que un IDE puede no ser compatible al recorrer el código del script durante la depuración.

Aplicando estilo a la aplicación con CSS

La última tarea es hacer que la aplciación de Login sea atractiva, añadiendo una hoja de estilos en cascada (CSS).

1. Cree una hoja de estilos

    a) En la ventana del proyecto, haga clic derecho en la carpeta fxmlexample debajo de Source Packages y seleccione **Nuevo** y, a continuación **Otros**.

    b) En el cuadro de diálogo **Nuevo archivo**, seleccione **Otro** y, a continuación **Cascading Style Sheet** y haga clic en **Siguiente**.

    c) Llamelo **Login** y haga clic en **Finalizar**.

    d) Copie los contenidos del archivo Login.css en su nuevo archivo de CSS. El archivo Login.css está incluido en el archivo LoginCSS.zip descargable. Para una descripción de las clases en el archivo CSS, consulte Formularios con JavaFX CSS.

2. Descargue la imagen gris, similar al lino para el fondo haciendo clic derecho en el archivo background.jpg y guardarlo en la carpeta fxmlexample.

3. Abra el archivo fxml_example.fxml y añada un elemento de estilo antes de que finalice el marckup en el layout GridPane como se muestra en el ejemplo 6-8.

    Hoja de estilo Ejemplo 6-8

    ```
    <stylesheets>
        <URL value="@Login.css"/>
    </stylesheets>

    </GridPane>
    ```

    El símbolo @ antes del nombre de la hoja de estilo en la dirección indica que la hoja de estilo se encuentra en el mismo directorio que el archivo FXML.

4. Para utilizar el estilo de raíz(root) para el panel de cuadrícula(GridPane), agregue una clase de estilo para el marckup del layout GridPane como se muestra en el ejemplo 6-9.

Ejemplo 6-9 Estilo del GridPane

```
<GridPane
fx:controller="fxmlexample.FXMLExampleController"
xmlns:fx=http://javafx.com/fxml
alignment="center"
hgap="10"
vgap="10"
styleClass="root"
>
```

5. Crear un ID para el objeto de texto “Welcome” por lo que utiliza el estilo #welcome-texto definido en el archivo CSS, como se muestra en el ejemplo 6-10.

Ejemplo 6-10 Texto ID

```
<Text
id="welcome-text"
text="Welcome"
GridPane.columnIndex="0"
GridPane.rowIndex="0"
GridPane.columnSpan="2"
/>
```

6. Ejecute la aplicación. La figura 6-5 muestra la aplicación estilizada. Si se encuentra con problemas, a continuación, eche un vistazo a el código que se incluye en el archivo descargable FXMLExample.zip.

Figura 6-5 Login estilizado

A donde ir ahora?

Ahora que usted está familiarizado con FXML, mire Introducción a FXML, que proporciona más información sobre los elementos que componen el lenguaje FXML. El documento está incluido en el paquete javafx.fxml en la documentación de la API.

También puede probar la herramienta JavaFX Scene Builder abriendo el archivo fxml_example.fxml en Scene Builder y hacer modificaciones. Esta herramienta proporciona un entorno de diseño visual para el diseño de la interfaz de usuario para las aplicaciones JavaFX y genera

automáticamente el código FXML para el diseño. Tenga en cuenta que el archivo FXML podría volver a formatear cuando se guardan. Véase la Guía de introducción a JavaFX Scene Builder para obtener más información sobre esta herramienta. El Skinning con CSS y la sección CSS Analizador de la Guía del usuario de escena JavaFX Builder también le brinda información sobre cómo puede la carcasa de su diseño FXML.

Animación y Efectos Visuales en JavaFX

Usted puede utilizar JavaFX para desarrollar rápidamente aplicaciones con experiencias de usuario ricas. En este Getting Started tutorial, usted aprenderá a crear objetos animados y lograr efectos complejos con muy poca codificación.

La figura 7-1 muestra la aplicación que se creará.

Figura 7-1 Colorful Circles Aplicación

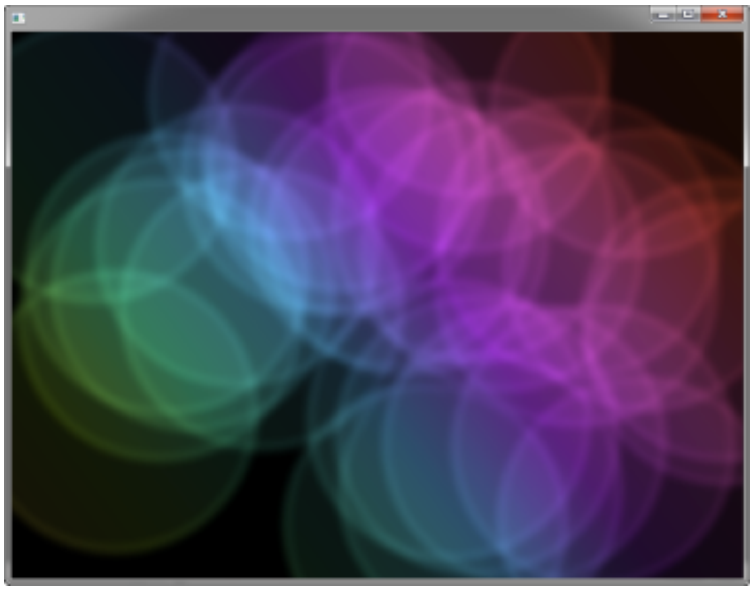
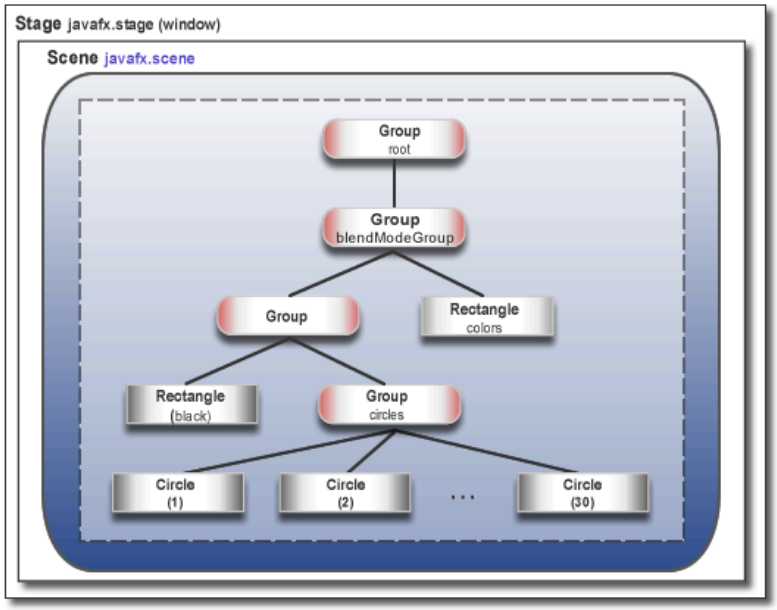


Figura 7-2 muestra el gráfico de la escena(Scene Graph) para la aplicación "ColorfulCircles". Los Node que raman son instancias de la clase de Group, y los nodos no enramados también conocidos como nodos hoja, son instancias de la clase Rectangle y Circle.

Figura 7-2 Círculos Coloridos Escena Gráfica(Scene Graph)



La herramienta utilizada en este tutorial de introducción es NetBeans IDE. Antes de empezar, asegúrese de que la versión de NetBeans IDE que está usando admite JavaFX 8. Consulte la sección Configuración de sistemas certificados de la página de Java SE Descargas para más detalles.

Configurando nuestra Aplicación

Configure su proyecto JavaFX en NetBeans IDE como sigue acontinuación:

1. En el menú **Archivo**, seleccione **Nuevo proyecto**.

2. En la categoría de aplicaciones JavaFX, elija **Aplicación JavaFX**. Haga clic en **Siguiente**.

3. Nombre a el proyecto: **ColorfulCircles** y haga clic en **Finalizar**.

4. Elimine las declaraciones de importación que NetBeans IDE genera.

   Puede generar las declaraciones de importación como usted trabaja a su manera a través de la guía de aprendizaje mediante el uso de cualquiera de la función de autocompletado de código o el comando Importaciones Fix en el menú Fuente en NetBeans IDE. Cuando hay una selección de declaraciones de importación, elija el que comienza con JavaFX.

Configurando el Proyecto

Elimine el contenido de la clase ColorfulCircles desde el archivo de origen que NetBeans IDE genera y reemplazarlo con el código en el ejemplo 7-1.

Ejemplo 7-1 Aplicación Básica

```
public class ColorfulCircles extends Application
{

    @Override
    public void start(Stage primaryStage)
    {
        Group root = new Group();
        Scene scene = new Scene(root, 800, 600, Color.BLACK);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

Para la aplicación **ColorfulCircles**, es conveniente utilizar un nodo de grupo como el nodo raíz(root) para la escena. El tamaño del grupo es dictado por el tamaño de los nodos dentro de ella. Para la mayoría de aplicaciones, sin embargo, desea que los nodos para realizar el seguimiento del tamaño de la escena y cambiar cuando se cambia el tamaño del escenario. En ese caso, se utiliza un nodo de layout de tamaño variable como la raíz(root), como se describe en Creación de un formulario en JavaFX.

Puede compilar y ejecutar la aplicación ColorfulCircles ahora, y en cada paso del tutorial, para ver los resultados intermedios. Si se encuentra con problemas, a continuación, eche un vistazo a el código en el archivo ColorfulCircles.java, que se incluye en el archivo ColorfulCircles.zip descargable. En este punto, la aplicación es una simple ventana en negro.

Agregando Graficos

A continuación, cree 30 círculos con el código en el ejemplo 7-2 antes de la línea `primaryStage.show()`.

Ejemplo 7-2 30 Círculos

```
Group circles = new Group();
for (int i = 0; i < 30; i++)
{
   Circle circle = new Circle(150, Color.web("white", 0.05));
   circle.setStrokeType(StrokeType.OUTSIDE);
   circle.setStroke(Color.web("white", 0.16));
   circle.setStrokeWidth(4);
   circles.getChildren().add(circle);
}
root.getChildren().add(circles);
```

Este código crea un grupo llamado “`circles`”, luego utiliza un bucle `for` para añadir 30 círculos a el grupo. Cada círculo tiene un radio de 150px, color de relleno de blanco, y el nivel de opacidad de un 0.5%, lo que significa que es casi transparente.

Para crear un borde alrededor de los círculos, el código incluye la clase StrokeType. Un tipo de trazo de FUERA (`OUTSIDE`) significa el límite del círculo se extiende fuera del interior por el valor `strokeWidth`, que es 4. El color del stroke es de color blanco, y el nivel de opacidad es 0.16%, lo que es más brillante que el color de los círculos.

La línea final agrega el grupo de círculos para el nodo raíz(root). Esta es una estructura temporal. Más tarde, modificará este escenario gráfico(Scene Graph) para que coincida con la que se muestra en la Figura 7-2.

La figura 7-3 muestra la aplicación. Debido a que el código no ha especificado todavía un lugar único para cada círculo, los círculos se dibujan en la parte superior de uno al otro, con la esquina superior

izquierda de la ventana, como el punto central para los círculos. La opacidad de los círculos superpuestos interactúa con el fondo negro, produciendo el color gris de los círculos .

Figura 7-3 Circles

Agregando un efecto Visual

Continuamos aplicando un efecto de desenfoque de cuadro (box blur effect) a los círculos de modo que aparezcan ligeramente fuera de foco. El código es en el Ejemplo 7-3. Agregue este código antes de la línea `primaryStage.show()`.

Ejemplo 7-3 Box Blur Effect

```
circles.setEffect(new BoxBlur(10, 10, 3));
```

Este código establece el radio de desenfoque a 10 píxeles de ancho por 10 píxeles de alto, y la iteración de desenfoque a 3, por lo que es la aproximación de un desenfoque gaussiano(Gaussian blur). Esta técnica desenfoque produce un efecto de suavizado en el borde de los círculos, como se muestra en la Figura 7-4.

Figura 7-4 Cuadro de la falta de definición en los círculos

Crear un degradado de fondo

Ahora, cree un rectángulo y lo rellenamos con un gradiente lineal, como se muestra en el ejemplo 7-4.

Agregue el código antes de que los `root.getChildren().add(circles)` de la línea para que el rectángulo degradado aparece detrás de los círculos.

Ejemplo 7-4 Linear Gradient

```
Rectangle colors = new Rectangle(scene.getWidth(), scene.getHeight(),
     new LinearGradient(0f, 1f, 1f, 0f, true, CycleMethod.NO_CYCLE, new
         Stop[]{
            new Stop(0, Color.web("#f8bd55")),
            new Stop(0.14, Color.web("#c0fe56")),
            new Stop(0.28, Color.web("#5dfbc1")),
            new Stop(0.43, Color.web("#64c2f8")),
            new Stop(0.57, Color.web("#be4af7")),
            new Stop(0.71, Color.web("#ed5fc2")),
            new Stop(0.85, Color.web("#ef504c")),
            new Stop(1, Color.web("#f2660f")),}));
colors.widthProperty().bind(scene.widthProperty());
colors.heightProperty().bind(scene.heightProperty());
root.getChildren().add(colors);
```

Este código crea un rectángulo llamado colores. El rectángulo es la misma anchura y altura que la escena y se llena con un gradiente lineal que comienza en la esquina inferior izquierda (0, 1) y termina en la esquina derecha superior (1, 0). El valor la verdad es proporcional al rectángulo y NO_CYCLE indica que el ciclo de color no se repetirá. La secuencia de parada[] indica lo que el degradado de color debe estar en un lugar en particular.

Las siguientes dos líneas de código hacen que el gradiente lineal de ajuste como el tamaño de los cambios de escena por la unión de la anchura y la altura del rectángulo con el ancho y el alto de la escena. Consulte Uso de Propiedades de JavaFX y enlaces para obtener más información sobre la unión.

La última línea de código agrega los colores rectángulo al nodo raíz.

Los círculos de color gris con los bordes borrosos aparecen ahora en la parte superior de un arco iris de colores, como se muestra en la Figura 7-5.

Figura 7-5 Linear Gradient

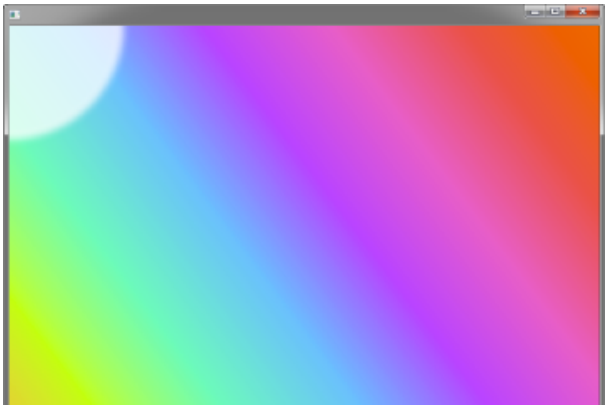
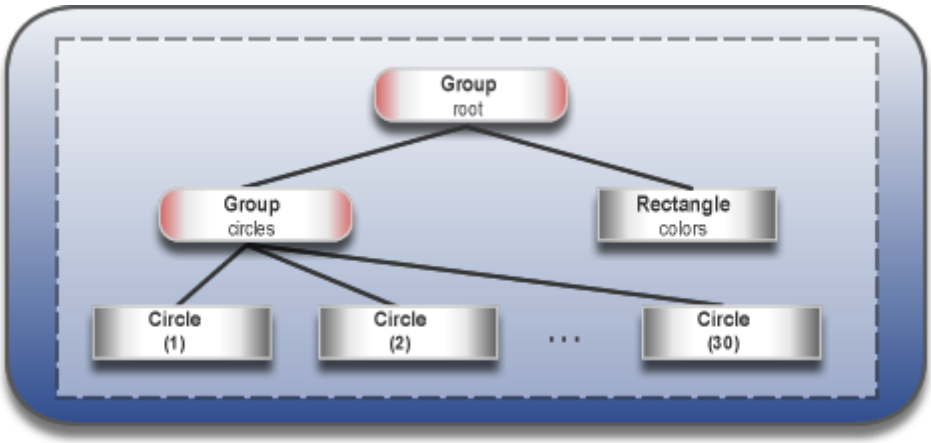


Figura 7-6 muestra el gráfico de la escena(Scene Graph) intermedia. En este punto, el grupo de círculos y colores rectángulo son hijos del nodo raíz.

Escena Figura 7-6 Gráfica(Scene Graph) Intermedio



Aplicar un modo de fusión(Blend Mode)

A continuación, añada color a los círculos y oscurezca la escena añadiendo un efecto de mezcla de superposición. Va a eliminar el grupo círculos y el rectángulo gradiente lineal desde el escenario gráfico y agregarlos al nuevo grupo de mezcla de superposición.

1. Busque las siguientes dos líneas de código:

   ```
   root.getChildren().add(colors);
   root.getChildren().add(circles);
   ```

2. Vuelva a colocar las dos líneas de código del paso 1 con el código en el ejemplo 7-5.

   Ejemplo 7-5 Modo de mezcla

   ```
   Group blendModeGroup =
       new Group(new Group(new Rectangle(scene.getWidth(), scene.getHeight(),
           Color.BLACK), circles), colors);
   colors.setBlendMode(BlendMode.OVERLAY);
   root.getChildren().add(blendModeGroup);
   ```

El grupo blendModeGroup configura la estructura de la mezcla de superposición. El grupo está formado por dos hijos. El primer hijo es un nuevo grupo (sin nombre) que contiene un nuevo (sin nombre) rectángulo negro y el grupo de círculos previamente creada. El segundo hijo es el rectángulo de colores creado previamente.

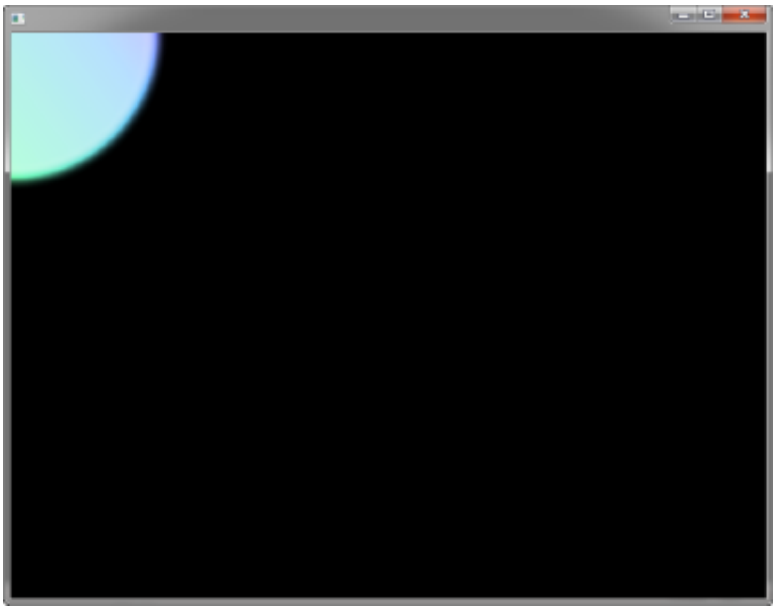
El método `setBlendMode()` se aplica la mezcla de superposición de colores del rectángulo. La última línea de código agrega el `blendModeGroup` al

escenario gráfico(SceneGraph) como un hijo del nodo raíz(node root), como se muestra en la Figura 7-2.

Una mezcla(blend) de superposición(overlay) es un efecto común en aplicaciones de diseño gráfico. Tal mezcla(blend) puede oscurecer una imagen o añadir luces o ambos, dependiendo de los colores en la mezcla(blend). En este caso, el rectángulo gradiente lineal se utiliza como la superposición(overlay). El rectángulo negro sirve para mantener el fondo oscuro, mientras que los círculos casi transparentes recogen los colores del degradado, sino que también se oscurecen.

La figura 7-7 muestra los resultados. Usted verá el efecto total de la mezcla de superposición cuando se anima a los círculos en el paso siguiente .

Figura 7-7 Overlay Blend



Añadir animación

El paso final es utilizar animaciones JavaFX para mover los círculos:

Si no lo ha hecho, añada importación java.lang.Math.random estática; a la lista de declaraciones de importación.

Agregue el código de animación en el Ejemplo 7-6 antes de la línea `primaryStage.show()`.

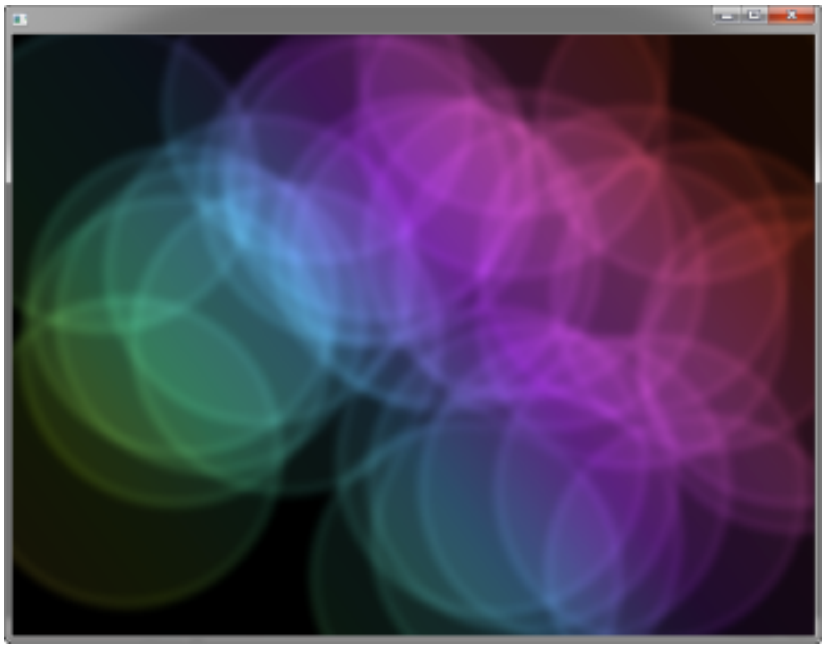
Ejemplo 7-6 Animación

```
Timeline timeline = new Timeline();
for (Node circle: circles.getChildren())
{
    timeline.getKeyFrames().addAll(
        new KeyFrame(Duration.ZERO, // set start position at 0
            new KeyValue(circle.translateXProperty(), random() * 800),
            new KeyValue(circle.translateYProperty(), random() * 600)
        ),
        new KeyFrame(new Duration(40000), // set end position at 40s
            new KeyValue(circle.translateXProperty(), random() * 800),
            new KeyValue(circle.translateYProperty(), random() * 600)
        )
    );
}
// play 40s of animation
timeline.play();
```

La animación es controlada por una línea de tiempo(`timeline`), por lo que este código crea una línea de tiempo(`timeline`), a continuación, utiliza un bucle `for` para sumar dos fotogramas(`frames`) claves, para cada uno de los 30 círculos. El primer fotograma clave en 0 segundos utiliza las propiedades `translateXProperty` y `translateYProperty` para establecer una posición aleatoria de los círculos dentro de la ventana. El segundo fotograma clave en 40 segundos hace lo mismo. Por lo tanto, cuando se reproduce la línea de tiempo, se anima a todos los círculos de una posición al azar a otro durante un período de 40 segundos.

La Figura 7-8 muestra los 30 círculos de colores en movimiento, lo que completa la solicitud. Para obtener el código fuente completo, consulte el archivo ColorfulCircles.java, que se incluye en el archivo descargable ColorfulCircles.zip…

Círculos Figura 7-8 animados



¿A dónde ir desde aquí?

Aquí hay varias sugerencias sobre qué hacer a continuación:

- Pruebe los ejemplos de JavaFX, que se puede descargar desde la sección de la página de Descargas de Java SE JDK Demos y Muestras en http://www.oracle.com/technetwork/java/javase/downloads/. Especialmente echar un vistazo a la aplicación Ensemble, que proporciona código de ejemplo para las animaciones y efectos.

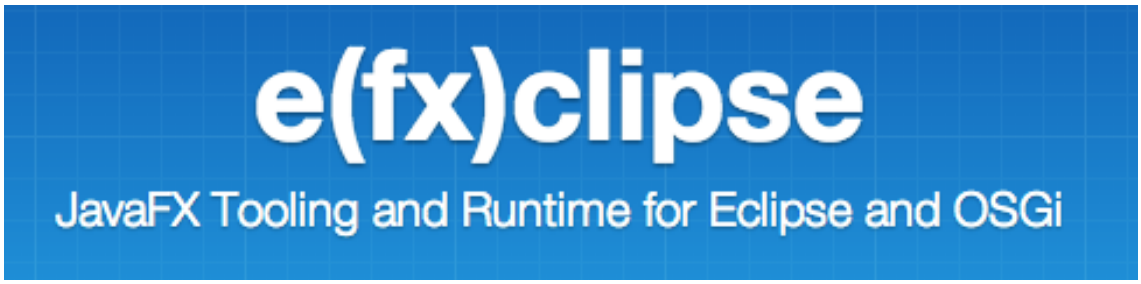
- Lea Creación de transiciones y Cronograma de Animación en JavaFX. Encontrará más detalles sobre la línea de tiempo de animación, así como información sobre la decoloración, camino, paralelo y transiciones secuenciales.

- Consulte Creación de Efectos Visuales de JavaFX para adicionales maneras de mejorar el aspecto y el diseño de la aplicación, incluyendo la reflexión, la iluminación y efectos de sombra.

- Pruebe la herramienta JavaFX Scene Builder para crear aplicaciones visualmente interesantes. Esta herramienta proporciona un entorno de diseño visual para el diseño de la interfaz de usuario para las aplicaciones JavaFX y genera código FXML. Usted puede utilizar el panel Propiedades o la opción Modificar de la barra de menú para añadir efectos a los elementos de la interfaz. Ver el Panel de propiedades y las secciones de la barra de menús de la Guía del usuario JavaFX Scene Builder para mas información.

Herramientas utilizadas y recomendadas:

0. JRE
1. JDK
2. JavaFX Scene Builder 1.1
3. NetBeans ó Eclipse(o el IDE que más te guste).
4. Eclipse(La Versión que prefieras, en este caso utilice Keppler).
5. e(fx)clipse plugin for eclipse

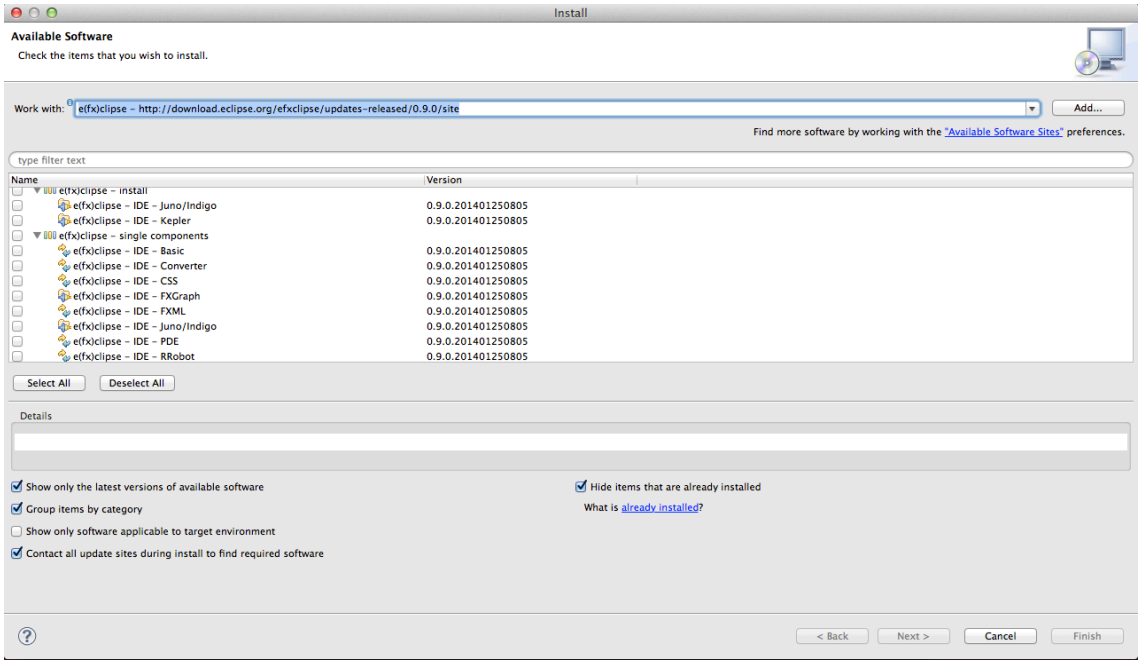


Plugin para el IDE eclipse, permite crear proyectos de JavaFX y tiene muy buena integración con el JavaFX CSS

Sitio web:
http://www.eclipse.org/efxclipse/index.html

URL para instalar:
http://download.eclipse.org/efxclipse/updates-released/0.9.0/site

Ejemplo:



6. Introducción a FXML: http://docs.oracle.com/javase/8/javafx/api/javafx/fxml/doc-files/introduction_to_fxml.html

7. Referencia de JavaFX CSS: http://docs.oracle.com/javase/8/javafx/api/javafx/scene/doc-files/cssref.html

8. Documentación Api's de JavaFX 8: http://docs.oracle.com/javase/8/javafx/api/toc.htm

9. Tecnologías, tutoriales oficiales de Oracle sobre FX: http://docs.oracle.com/javase/8/index.html

10. FXExperience buen blog sobre JavaFX(ingles!): http://fxexperience.com/

11. GuiGarage otro buen blog sobre JavaFX(ingles!): http://www.guigarage.com/

12. Proyecto de lujo para bases de dates en tiempo real y con componentes JavaFX, http://www.javafxdata.org/

13. Proyecto para pruebas unitarias especialmente para JavaFX: https://github.com/SmartBear/TestFX

14. Api de DataFX: http://datafx.bitbucket.org/datafx-core/

15. RovoBM, crea aplicaciones para iOS con JavaFX: http://www.robovm.org/

16. Dejo un proyecto que he realizado en eclipse, en el cual muestro como crear simples login de usuario sin css, con css, en un solo archivo, alguno por ahí con mvc y otro que muestra el acceso a otra ventana ó genera dialogos.

Deje swing hace días porque me limitaba a tener que entender sus apis desactualizadas, un ejemplo también muy claro es no usar frameworks como Java Media Frameworks porque con JavaFX tienes efectos, sombras, transciciones, Animaciones 3D.

Según en su página oficial todo el proyecto JavaFX ya no es Beta, esta en pañales pero existen muchos proyectos que ni por cerca Swing llego a crear, los invito a que empiecen en este nuevo framework y crezca esta comunidad.

Desde Honduras un abrazo.