1. Refuerzo de Java

Anexo IV. Animaciones en JavaFX

Programación de Servicios y Procesos

Arturo Bernal
Nacho Iborra
Javier Carrasco

Tabla de contenido

1. Animaciones en JavaFX

JavaFX también proporciona algunas formas de animar los controles o nodos. Básicamente, lo que se hará es cambiar las propiedades de los nodos (tamaño, posición, color, …) a intervalos regulares. En este proceso, hay dos factores importantes:

- ¿Cuánto se cambia las propiedades del nodo? Por ejemplo, si se está moviendo un botón, se moverá más rápido si se aumenta su posición X de 3 en 3 píxeles que si la aumentamos de 1 en 1 píxel.
- La frecuencia de estos cambios. Si se cambia la posición del botón cada 20 milisegundos, se moverá más lento que si se cambia cada 10 milisegundos.

Hay dos formas de hacer una animación: una de ellas es totalmente manual y la otra es (casi) totalmente automática.

1.1. Animaciones automáticas

La forma más sencilla de crear una animación es mediante el uso de clases de transición. Hay algunos tipos de clases de transición, como ***TranslateTransition*** (mover elementos), ***RotateTransition*** (rotar elementos), ***FadeTransition*** (para cambiar la opacidad), … Todas estas transiciones tienen algunos métodos útiles, como:

- ***play / playFromStart***: reproduce la animación desde su posición actual o desde el principio, respectivamente
- ***stop / pause***: detiene o pausa la animación, respectivamente.
- ***setCycleCount***: establece cuántas veces debe repetirse la animación. Si quieres repetirlo indefinidamente, puedes utilizar la constante *Timeline.INDEFINITE*.
- ***setAutoReverse***: establece si la animación debe revertirse cada vez que finaliza.
- ***setInterpolator***: establece cómo calcular las posiciones intermedias de la animación. Algunos de sus posibles valores son *Interpolator.LINEAR* (siempre a la misma velocidad), *Interpolator.EASE_IN* (más rápido al principio), *Interpolator.EASE_OUT* (más rápido al final), *Interpolator.EASE_BOTH* (valor predeterminado, significa rápido al principio y al final), …

Aparte de estos métodos, también existen algunos métodos particulares que pertenecen a cada tipo concreto de transición. Por ejemplo, si utilizas un *FadeTransition*, tendrás que establecer el valor inicial de la opacidad y el valor final. Si utilizas *RotateTransition*, deberás especificar los ángulos inicial y final.

El siguiente ejemplo mueve una etiqueta desde la parte izquierda de la ventana a la parte derecha (asumiendo es una ventana de 300px de ancho), en 3 segundos.

```
@FXML
private Label lbl;
…
    // Inside some method:
    TranslateTransition t = new TranslateTransition(Duration.millis(3000), lbl);
    t.setFromX(0);
    t.setFromY(0);
    t.setToX(300);
    t.setToY(0);
    t.play();
…
```

En el constructor de transición, se especifica la duración de la transición (3 segundos) y el nodo, o control, al que se aplicará (la etiqueta). Luego, se establecen los valores inicial y final (posición) del nodo (etiqueta), y el objeto *TranslateTransition* toma el control. En este ejemplo, se podría haber omitido las líneas que hacen referencia a la coordenada Y (métodos *t.setFromY* y *t.setToY*), ya que se está moviendo la etiqueta horizontalmente.

También se pueden combinar algunas transiciones para ejecutar una tras otra utilizando la clase ***SequentialTransition***. Se pueden añadir tantas transiciones como se necesiten en su constructor, y por último se llama al método ***play()***.

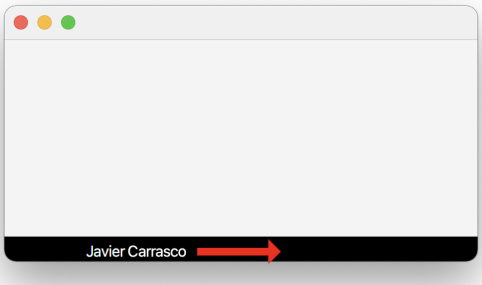
```
SequentialTransition s = new SequentialTransition(t1, t2, t3, t4);
s.play();
```

Si se necesita ejecutar algunas transiciones en paralelo (simultáneamente), se utilizará la clase ***ParallelTransition***, de la misma manera que *SequentialTransition*:

```
ParallelTransition p = new ParallelTransition(t1, t2);
p.play();
```

Ejercicio 1

Crea un proyecto llamado ***AnimationCredits***. La ventana tendrá un *FlowPane* de color negro en su parte inferior (utiliza un *BorderPane*), y dentro de esto *FlowPane* habrá una etiqueta con texto blanco con tu nombre. Este texto debe aparecer desde el lado izquierdo de la ventana y esconderse por el lado derecho continuamente.



1.2. Animaciones manuales

Si prefieres crear una animación de forma manual, necesitarás trabajar con las siguientes clases:

- ***KeyFrame***, es un intervalo de tiempo que lanzará un *ActionEvent* cada vez que expire. Es necesario especificar la duración del intervalo y el controlador de eventos que manejará el *ActionEvent*.
- ***TimeLine***, secuencia de *KeyFrames*. Cuando se llama, cada *KeyFrame* se reproduce secuencialmente.

Si quieres crear la misma animación que en el ejemplo anterior con estas clases, el código sería el siguiente:

```
@FXML
private Label lblName;
…
// Inside some method:
KeyFrame kf = new KeyFrame(Duration.millis(10), e -> {
    lbl.setTranslateX(lbl.getTranslateX() + 1);
});

Timeline t = new Timeline(kf);
t.setCycleCount(300);
t.play();
…
```

En primer lugar, se define la etiqueta y se añade al panel de diseño, como en el ejemplo anterior. Entonces, se define el *KeyFrame*. Se activará cada 10 milisegundos y luego aumentará la coordenada X de la etiqueta en 1 píxel. Finalmente, se añade el *KeyFrame* a un *TimeLine*, y se establece el recuento de ciclos. Este valor debe calcularse manualmente. En este ejemplo, si quieres mover la etiqueta de X = 0 a X = 300 en 3 segundos, moviéndola 1 píxel cada 10 milisegundos, entonces se necesitan un total de 300 iteraciones para llegar a X = 300.

Ejercicio 2

Crea un proyecto llamado ***GrowingButton*** con un botón en el medio (utiliza un *BorderPane*), con un ancho predefinido de 100 píxeles y una altura de 50 píxeles (usa las propiedades correspondientes, o los métodos *setPrefWidth()* y *setPrefHeight()* para establecer estos valores al inicio). Cada vez que se mueva el ratón dentro del botón, su ancho y alto deberán crecer hasta 300 x 250 píxeles, en 2 segundos. En cuanto salga el ratón fuera del botón, deberá recuperar su tamaño inicial. Utiliza la técnica manual para reproducir la animación.