Eventos en JavaFX

El documento contiene ejemplos de manejadores para distintas clases de eventos de JavaFX, los manejadores se muestran como métodos de conveniencia y también con addEventHandler.

La siguiente tabla contiene algunos eventos que se generan cuando el usuario interactúa con la interfaz de usuario.

Acción de usuario	Objeto fuente	Evento generado	Método de conveniencia con manejador anónimo
Click en botón	Button	ActionEvent	`setOnAction(new EventHandler<ActionEvent>() {..})`
Enter sobre campo texto	TextField	ActionEvent	`setOnAction(new EventHandler<ActionEvent>() {..})`
Marcar, desmarcar	RadioButton	ActionEvent	`setOnAction(new EventHandler<ActionEvent>() {..})`
Marcar, desmarcar	CheckBox	ActionEvent	`setOnAction(new EventHandler<ActionEvent>() {..})`
Seleccionar nuevo elemento	ComboBox	ActionEvent	`setOnAction(new EventHandler<ActionEvent>() {..})`
Ratón pulsado	Nodo, Escena	MouseEvent	`setOnMousePresed(new EventHandler<MouseEvent>() {..})`
Ratón soltado	Nodo, Escena	MouseEvent	`setOnMouseReleased(new EventHandler<MouseEvent>() {..})`
Click Ratón	Nodo, Escena	MouseEvent	`setOnMouseClicked(new EventHandler<MouseEvent>() {..})`
Ratón entró	Nodo, Escena	MouseEvent	`setOnMouseEntered(new EventHandler<MouseEvent>() {..})`
Ratón movido	Nodo, Escena	MouseEvent	`setOnMouseMoved(new EventHandler<MouseEvent>() {..})`
Ratón salió	Nodo, Escena	MouseEvent	`setOnMouseExited(new EventHandler<MouseEvent>() {..})`
Ratón arrastrado	Nodo, Escena	MouseEvent	`setOnMouseDragged(new EventHandler<MouseEvent>() {..})`
Tecla presionada	Nodo, Escena	KeyEvent	`setOnKeyPressed(new EventHandler<KeyEvent>() {..})`
Tecla soltada	Nodo, Escena	KeyEvent	`setOnKeyReleased(new EventHandler<KeyEvent>() {..})`
Tecla pulsada (i.e presionada y soltada)	Nodo, Escena	KeyEvent	`setOnKeyTyped(new EventHandler<KeyEvent>() {..})`

Ejemplo <Enter> sobre un TextField

Para el caso de introducir <Enter> sobre el campo de texto, el código que lo gestiona se explica a continuación. Recuerde que debemos inyectar el campo de texto en el controlador (@FXML) y en la inicialización del mismo añadir al campo de texto el método de conveniencia setOnAction.

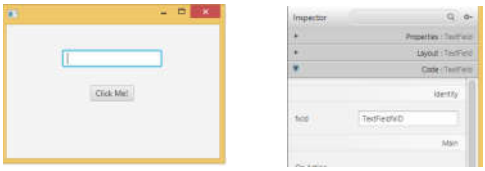


Figura 1 Interfaz para el ejemplo y vista de Scene Builder

En el proyecto que genera por defecto NetBeans, añadimos el TextField y definimos en Code fx:id TextfieldfxID.

```
@FXML  private TextField TextFieldfxID;

@FXML
    public void initialize() {
        //
        TextFieldfxID.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                System.out.println("Pulsado enter sobre el campo de texto");
            }});
    }
```

Figura 2 Método de conveniencia para capturar <enter> sobre un TextField

Lo mismo puede conseguirse si en lugar del método de conveniencia usamos en la inicialización el método `addEventHandler`. Recuerde que puede utilizar indistintamente el método de conveniencia o el método que añade el manejador de evento.

```
@FXML
public void initialize() {
        //
TextFieldfxID.addEventHandler(ActionEvent.ACTION, new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                System.out.println("Pulsado enter sobre el campo de texto");
            }
        });

}
```

Figura 3 Manejador de evento para el TextField, detectar <enter>

En cualquier caso tanto la figura 2 como la 3 pueden simplificarse utilizando lambda expresiones.

```
// Para la figura 2
@FXML
public void initialize() {

TextFieldfxID.setOnAction((ActionEvent event) -> {
            System.out.println("Pulsado enter sobre el campo de texto");
        });

}

// Para la figura 3
@FXML
public void initialize() {
TextFieldfxID.addEventHandler(ActionEvent.ACTION, (ActionEvent event) -> {
            System.out.println("Pulsado enter sobre el campo de texto");
        });

}
```

Figura 4 Código con lambda expresiones equivalentes al de las figuras 2 y 3

NetBeans le permite hacer automáticamente esa sustitución pulsando con el botón derecho sobre la pequeña bombilla amarilla en la vista de edición de código.

Eventos de teclado

Sobre el campo de texto del ejemplo precedente vamos a definir ahora un oyente de teclado que capture los 3 tipos de eventos de teclado (`KeyPressed, KeyReleased, KeyTyped`). Si en tiempo de ejecución tiene el foco entonces puede recibir los eventos de teclado, si la ventana está minimizada no se captura nada.

Tenemos ahora que poner el siguiente código en el método `initialize` del controlador.

```
TextFieldfxID.addEventHandler(KeyEvent.ANY, new EventHandler<KeyEvent>() {
 @Override
 public void handle(KeyEvent event)
{
System.out.println("Clase del evento de teclado: " + event.getClass());
System.out.println("Tipo del evento de teclado: " + event.getEventType().getName());
System.out.println("Código de la tecla: " + event.getCode() +  " Texto asociado: " + event.getText());
}
        });
```

Figura 5 Captura de cualquier evento de teclado sobre el TextField

El código anterior intercepta todos los eventos de teclado. Si presionamos y soltamos la tecla por ejemplo “1”, el eco en pantalla es el siguiente:

```
Clase del evento de teclado: class javafx.scene.input.KeyEvent
Tipo del evento de teclado: KEY_PRESSED
Código de la tecla: DIGIT1 Texto asociado: 1
Clase del evento de teclado: class javafx.scene.input.KeyEvent
Tipo del evento de teclado: KEY_TYPED
Código de la tecla: UNDEFINED Texto asociado:
Clase del evento de teclado: class javafx.scene.input.KeyEvent
Tipo del evento de teclado: KEY_RELEASED
Código de la tecla: DIGIT1 Texto asociado: 1
```

Observe la secuencia de eventos generados al pulsar y soltar es: KEY_PRESSED, KEY_TYPED y KEY_RELEASED.

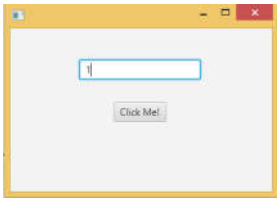


Figura 6 Captura de ejecución al introducir “1” sobre el campo de texto

Todas las teclas no tienen el mismo comportamiento. Las teclas de desplazamiento (las 4 flechas) y las teclas de función (F1,..,F10) al pulsarlas y soltarlas generan el siguiente eco en pantalla:

```
Clase del evento de teclado: class javafx.scene.input.KeyEvent
Tipo del evento de teclado: KEY_PRESSED
Código de la tecla: UP Texto asociado:
Clase del evento de teclado: class javafx.scene.input.KeyEvent
Tipo del evento de teclado: KEY_RELEASED
Código de la tecla: UP Texto asociado:
```

No generan el evento KEY_TYPED.

Volviendo al ejemplo de la captura de eventos de teclado en un TextField, si quisiésemos capturar únicamente el evento KEY_TYPED, el método anterior debe ponerse en la siguiente forma:

```
TextFieldfxID.addEventHandler(KeyEvent.KEY_TYPED, new EventHandler<KeyEvent>() {
    @Override
    public void handle(KeyEvent event) {
        System.out.println("Clase del evento de teclado: " + event.getClass());
        System.out.println("Tipo del evento de teclado: " + event.getEventType().getName());
        System.out.println("Código de la tecla: " + event.getCode() +  " Texto asociado: " + event.getText());
    }
});
```

Figura 7 Código para capturar key_typed

Lo único que hemos cambiado es el primer parámetro de KeyEvent.ANY a KeyEvent.KEY_TYPED.

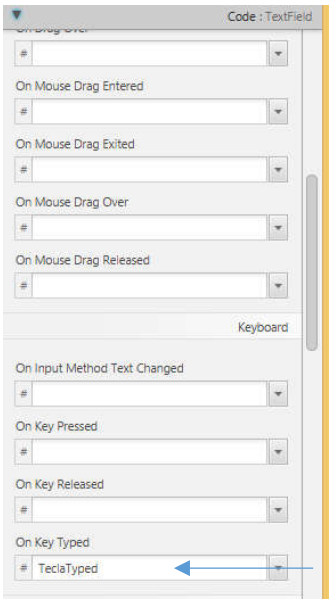
El mismo funcionamiento de la Figura 7 con un método de conveniencia en el método de inicialización del controlador.

```
TextFieldfxID.setOnKeyTyped(new EventHandler<KeyEvent>() {

        @Override
         public void handle(KeyEvent event)
        {
        System.out.println("Clase del evento de teclado: " + event.getClass());
        System.out.println("Tipo del evento de teclado: " + event.getEventType().getName());
        System.out.println("Código de la tecla: " + event.getCode() +  " Texto asociado: " + event.getText());
        }
        });
```

Figura 8 Método de conveniencia para capturar KEY_TYPED

Los ejemplos anteriores se pueden programar **de una tercera forma**, incluyendo un nombre de método dentro de Scene Builder en la pestaña de código. El código de la figura 7 u 8 puede eliminarse si en Scene Builder seleccionamos el componente TextField y en la pestaña código introducimos un nombre (TeclaTyped, el nombre puede ser cualquiera) en la entrada On Key Typed



Controlador

```
@FXML
void TeclaTyped(KeyEvent event) {
   System.out.println("Clase del evento de teclado: " + event.getClass());
   System.out.println("Tipo del evento de teclado: " +
event.getEventType().getName());
   System.out.println("Código de la tecla: " + event.getCode() +  " Texto
asociado: " + event.getText());

    }
```

Figura 9 Manejo de eventos inyectando un método en el controlador

Ahora no es necesario poner código en el método `inicialize` del controlador, en otras palabras no se añade explícitamente el manejador de evento.

Captura de combinaciones de teclas

Cuando se quiere capturar combinaciones de teclas, como por ejemplo CONTROL-R, o cualquier otra combinación, debe utilizarse un código especial. La captura puede ser sobre la escena o bien sobre el TextField que estamos usando como ejemplo. Cambie el código del método `initialize` por el que figura a continuación

```
final KeyCombination keyComb1 = new KeyCodeCombination(KeyCode.R, KeyCombination.CONTROL_DOWN);

TextFieldfxID.addEventHandler(KeyEvent.KEY_RELEASED, new EventHandler<KeyEvent>() {
           @Override
           public void handle(KeyEvent event) {
               if (keyComb1.match(event)) {
                        System.out.println("Ctrl+R pulsado y soltado");
                                           }
                                       }
                                   });
```

Figura 10 Código para detectar combinaciones de teclas sobre el campo de texto

Usamos la clase KeyCombination, donde al instanciar definimos la combinación de teclas, en este caso Control-R. Se añade un manejador en el método de inicialización del controlador, el cual detecta que la tecla R de la combinación ha sido soltada.

Eventos de ratón

Un evento de ratón se dispara o activa en las siguientes situaciones: un botón del ratón es presionado, un botón del ratón es soltado, se hace un click o doble click, se arrastra el ratón manteniendo pulsado uno de sus botones, el ratón entra en algún nodo de la escena, sale del mismo, etc.

La clase `MouseEvent` contiene la siguiente información

javafx.scene.input.MouseEvent	
+getButton(): MouseButton	Indicates which mouse button has been clicked.
+getClickCount(): int	Returns the number of mouse clicks associated with this event.
+getX(): double	Returns the *x*-coordinate of the mouse point in the event source node.
+getY(): double	Returns the *y*-coordinate of the mouse point in the event source node.
+getSceneX(): double	Returns the *x*-coordinate of the mouse point in the scene.
+getSceneY(): double	Returns the *y*-coordinate of the mouse point in the scene.
+getScreenX(): double	Returns the *x*-coordinate of the mouse point in the screen.
+getScreenY(): double	Returns the *y*-coordinate of the mouse point in the screen.
+isAltDown(): boolean	Returns true if the `Alt` key is pressed on this event.
+isControlDown(): boolean	Returns true if the `Control` key is pressed on this event.
+isMetaDown(): boolean	Returns true if the mouse `Meta` button is pressed on this event.
+isShiftDown(): boolean	Returns true if the `Shift` key is pressed on this event.

`MouseButton` define las siguientes constantes: `PRIMARY, SECONDART, MIDDLE, NONE`. Indican el botón que se ha pulsado.

- ANY: Es el tipo base de todos los tipos de eventos de ratón. Si queremos que un nodo reciba todos los eventos de ratón debemos registrar un manejador en el cual el primer parámetro es MouseEvent.ANY
- MOUSE_PRESSED: Al presionar un botón del ratón se genera este evento. El método getButton() indica qué botón se ha presionado.
- MOUSE_RELEASED: Al soltar un botón del ratón se genera este evento. El evento se entrega al mismo nodo en el cual el botón fue presionado.
- MOUSE_CLICKED: Generado al hacer click sobre un nodo. El botón del ratón tiene que ser pulsado y soltado para que se genere el evento.
- MOUSE_MOVED: Al mover el ratón sin presionar ningún botón se genera este evento.
- MOUSE_ENTERED: Se genera cuando el ratón entra en un nodo.
- MOUSE_EXITED: Se genera cuando el ratón abandona un nodo.
- DRAG_DETECTED: Se genera al presionar el ratón y arrastrar un nodo y recorrer cierta distancia que depende de la plataforma en la que se esté ejecutando el programa.
- MOUSE_DRAGGED: Al mover el ratón manteniendo un botón pulsado se genera el evento.

Ejemplo

Sobre la interfaz de la figura 1 añadiremos un círculo con el objetivo de definir manejadores de eventos de ratón.

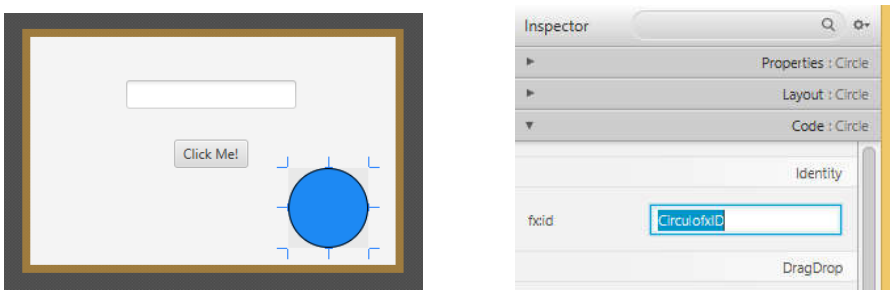


Figura 11 Interfaz para los eventos de ratón y definición de la identidad fx:id

Sobre la escena o bien sobre el círculo definiremos un manejador de eventos que detecte si algún botón del ratón se ha presionado.

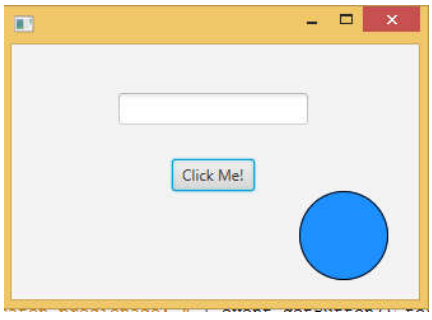
En la inicialización del controlador añadimos el código de manejo del evento.

```
@FXML   public void initialize() {

CirculofxID.addEventHandler(MouseEvent.MOUSE_PRESSED, new EventHandler<MouseEvent>() {
   @Override
   public void handle(MouseEvent event) {
     System.out.println("Botón de ratón presionado: " + event.getButton().toString());
        }
    });
}
```

Figura 12 Manejador de evento en el círculo para botón de ratón presionado

En ejecución, con el ratón podemos presionar sobre el círculo con el botón primario, secundario y sobre el central. La parte derecha de la figura muestra la captura en ejecución después de pulsar los tres botones.




```
Botón de ratón presionado: PRIMARY
Botón de ratón presionado: SECONDARY
Botón de ratón presionado: MIDDLE
```

Figura 13 Pulsación de los botones del ratón sobre el círculo

Para usar el método de conveniencia correspondiente lo único que tenemos que cambiar es la interfaz y el nombre del mismo.

```java
CirculofxID.setOnMousePressed(new EventHandler<MouseEvent>() {
@Override
public void handle(MouseEvent event) {
    System.out.println("Botón de ratón presionado: " + event.getButton().toString());

}

});
```

Figura 14 Método de conveniencia para detectar pulsaciones de los botones del ratón.

Si queremos capturar cualquier evento de ratón sobre el círculo tenemos que cambiar el manejador del evento de la figura 11 y usar `MouseEvent.ANY`.

Funciona igual la expresión: event.getEventType();

```java
@FXML
public void initialize() {

    CirculofxID.addEventHandler(MouseEvent.ANY, new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        System.out.println("Tipo del evento: " + event.getEventType().getName());
        System.out.println("Botón de ratón presionado: " + event.getButton().toString());

    }

});
```

Figura 15 Manejador sobre el círculo para todos los eventos de ratón equivalente a la figura 14

Una posible traza de ejecución se muestra a continuación.

```
Tipo del evento: MOUSE_ENTERED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_MOVED
Botón de ratón presionado: NONE
Tipo del evento: MOUSE_EXITED
Botón de ratón presionado: NONE
```

El usuario ha entrado con el ratón al círculo (MOUSE_ENTERED), lo ha movido (MOUSE_MOVED) varias veces y luego ha salido del círculo (MOUSE_EXITED).

Otra posible traza de ejecución puede contener los siguientes eventos:

```
Tipo de evento: MOUSE_MOVED
Tipo de evento: MOUSE_PRESSED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: DRAG_DETECTED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: MOUSE_DRAGGED
Tipo de evento: MOUSE_RELEASED
Tipo de evento: MOUSE_CLICKED
Tipo de evento: MOUSE_MOVED
Tipo de evento: MOUSE_MOVED
Tipo de evento: MOUSE_EXITED
```

En este caso ha movido el ratón (MOUSE_MOVED), ha presionado un botón (MOUSE_PRESSED) y manteniéndolo pulsado lo ha arrastrado (MOUSE_DRAGGED), cuando se ha arrastrado una distancia (que depende de la plataforma en la cual se esté ejecutando el programa) se genera el evento DRAG_DETECTED. Cuando se suelta el botón del ratón se genera MOUSE_RELEASED, y por último se ha hecho un click (MOUSE_CLICKED), se ha movido y ha salida de la figura.

Obviamente hace falta código para mover el círculo cuando se detecte que éste está siendo arrastrado.