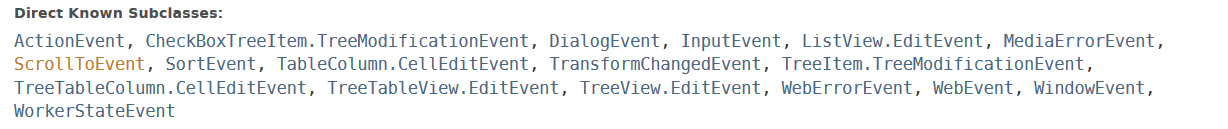
# Eventos javaFX

Los eventos en la **API JavaFX** son parte importante en el desarrollo de una aplicación, un evento se produce cuando el usuario interactúa con la aplicación, por ejemplo, al hacer clic sobre un botón, al mover el mouse sobre algún Node de la escena, al presionar una tecla, o al seleccionar un elemento de una lista, entre muchas otras cosas.

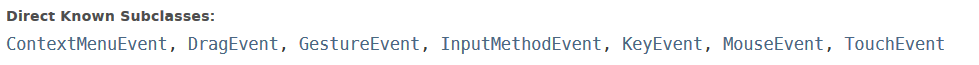
La clase base que representa los eventos es javafx.event.Event todas las subclases de la misma también representan eventos, algunas de estas clases son:

* **MouseEvent**: Representa los eventos producidos por la interacción del ratón, por ejemplo, hacer clic, mover el cursor, girar la rueda, etc.
* **KeyEvent**: Esta clase es para los eventos producidos por el teclado, puede ser, presionar o liberar un tecla.
* **WindowEvent**: Aquí tenemos los eventos producidos por la ventana, por ejemplo, el mostrar u ocultar la ventana.

**Subclases de Event**:



Como clases hijas de InputEvent tenemos:



Para controlar un evento en JavaFX disponemos de los manejadores de eventos y los filtros de eventos, más adelante veremos la diferencia entre ellos, cada uno de estos controladores poseen estas propiedades:

* **Target**: El nodo en donde se produjo el evento, un botón, la ventana, etc..
* **Source**: Indica la fuente que genera el evento, el teclado, el ratón, etc..
* **Type**: Se refiera el tipo de evento producido, presionar un botón del mouse, presionar una tecla, mover el mouse, etc..

Para responder a un evento requerimos un objeto que implemente la interface **EventHandler**, seguido mostramos dos maneras de crear este objeto, el primero usando clases anónimas y el segundo usa expresiones lambdas, este último requiere Java 8.

EventHandler<MouseEvent> handler1 = **new** EventHandler<MouseEvent>() {

@Override

**public** void handle(MouseEvent event) {

System.out.println("handler1...");

}

};

EventHandler<MouseEvent> handler2 = (MouseEvent event) -> {

System.out.println("handler2...");

};

Para agregar estos manejadores de eventos usaremos los métodos addEventHandler() o addEventFilter(), debemos indicar como parámetros, **el tipo de evento** que deseamos manejar, y el **manejador de eventos**, del mismo modo removemos o quitamos un manejador usando las funciones, removeEventFilter() o removeEventHandler().

Button btn = **new** Button();

btn.setText("Button");

btn.addEventHandler(MouseEvent.MOUSE\_CLICKED, handler1);

StackPane root = **new** StackPane();

root.getChildren().add(btn);

root.addEventFilter(MouseEvent.MOUSE\_CLICKED, handler2);

Scene scene = **new** Scene(root, 300, 250);

primaryStage.setTitle("JavaFX Manejo de Eventos");

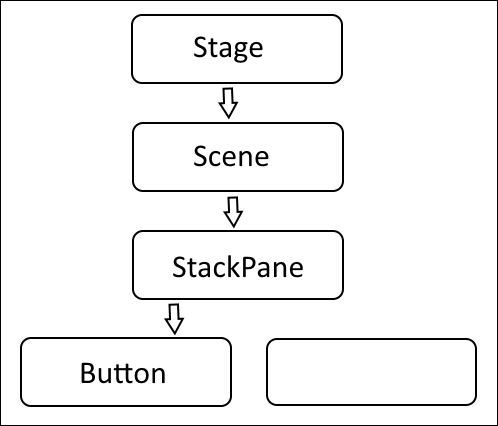
primaryStage.setScene(scene);

primaryStage.show();

Esta GUI corresponde a un Button que se encuentra dentro de un StackPane, al primero le agregamos un manejador de eventos, y al segundo un filtro de eventos, si ejecutamos y hacemos clic sobre el botón veremos que primero se llama al **handler2** y luego al **handler1**, podemos ver la consola de salida con los correspondientes mensajes.

Para entender lo que ocurre debemos saber que el proceso de generación de un evento tiene varias fases:

**Construcción de la ruta**: se crea la ruta que seguirá el evento, iniciando en la ventana (Stage) hasta el objeto en donde se origina el evento, para nuestro caso el Button.

[](https://lh3.googleusercontent.com/-9rhHoH_lrcg/WVEL-eFSJrI/AAAAAAAADIc/cxgWPxjJArQ2352fx1LI_ZHl0j5MqIVtwCHMYCw/s1600-h/eventos-javafx3)

**Fase de captura**: el nodo raíz distribuye el evento, dicho evento recorre la jerarquía iniciando en la parte superior, si alguno de los nodos de ha registrado un **filtro**, el mismo es invocado, esto ocurre hasta llegar a la parte inferior del recorrido que termina con el objeto que originó el evento.

**Fase de propagación**: ocurre el proceso inverso, el evento se distribuye iniciando el recorrido en el origen del evento hasta llegar al nodo raíz, si algún nodo ha registrado un manejador de eventos este será invocado, el proceso termina al llegar al nodo raíz.

Debemos saber que es posible establecer un manejador de evento usando su método **setter** respectivo, este método tiene la siguiente forma setOnXXX() donde XXX corresponde al tipo de evento para el cual deseamos agregar el controlador, por ejemplo, para agregar un manejador de evento para el clic sobre un botón:

Button btn = **new** Button("Button");

btn.setOnMouseClicked(event -> System.out.println("clic!"));

btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent event) {

                //Acciones a realizar al clicar el botón

            }

        });

Este es un ejemplo de eventos del teclado:

TextField tf1 = **new** TextField();

PasswordField tf2 = **new** PasswordField();

TextArea tf3 = **new** TextArea();

VBox root = **new** VBox(tf1, tf2, tf3);

root.setSpacing(5.0);

root.setPadding(**new** Insets(5.0));

Scene scene = **new** Scene(root, 300, 250);

scene.addEventFilter(KeyEvent.KEY\_TYPED, e -> {

String type = e.getEventType().getName();

String source = e.getSource().getClass().getSimpleName();

String target = e.getTarget().getClass().getSimpleName();

System.out.println("filter: " + type + ", " + source + ", " + target);

**if** (Character.isDigit(e.getCharacter().charAt(0))) {

System.out.println("caracter: " + e.getCharacter() + ", no permitido.");

e.consume();

}

});

primaryStage.setTitle("JavaFX Manejo de Eventos");

primaryStage.setScene(scene);

primaryStage.show();

Este pequeño código utiliza tres controles que nos sirven para introducir texto, lo que buscamos es demostrar cómo podemos utilizar los filtros de eventos para impedir que en cualquiera de los tres controles se introduzcan números.

Para lograr el objetivo agregamos un filtro para el evento que indica que se ha utilizado el teclado sobre nuestro Scene, luego comprobamos la tecla, si la misma corresponde a un digito rompemos el recorrido del evento usando el método cosume().

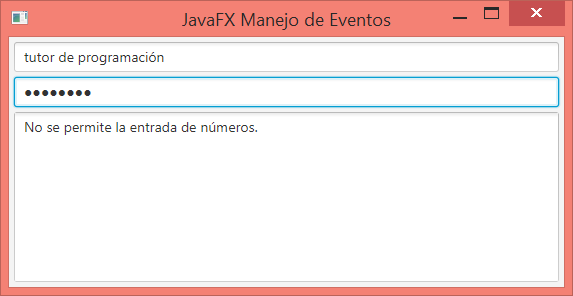
scene.addEventFilter(KeyEvent.KEY\_TYPED, e -> {

**if** (Character.isDigit(e.getCharacter().charAt(0))) {

e.consume();

}

});

[](https://lh3.googleusercontent.com/-WEF4GTiPC3I/WVEMANWgoHI/AAAAAAAADIk/Gcg4T0A7-SkB5KKYVom9FI9TAM6J_46vwCHMYCw/s1600-h/image5)

De este modo bloqueamos la escritura de números en cualquier campo de texto que esté dentro de nuestra escena.

**Asignación de eventos “método asociado”.**

Es el método usado hasta ahora. Asignamos un método para que reaccione a un evento concreto:

<Button fx:id="botonLimpiar" mnemonicParsing="false" **onAction="#limpiarContenidos"** text="Limpiar" />

|  |
| --- |
| @FXML  public void limpiarContenidos(){    //Acciones a realizar al clicar el botón.      } |
|  |