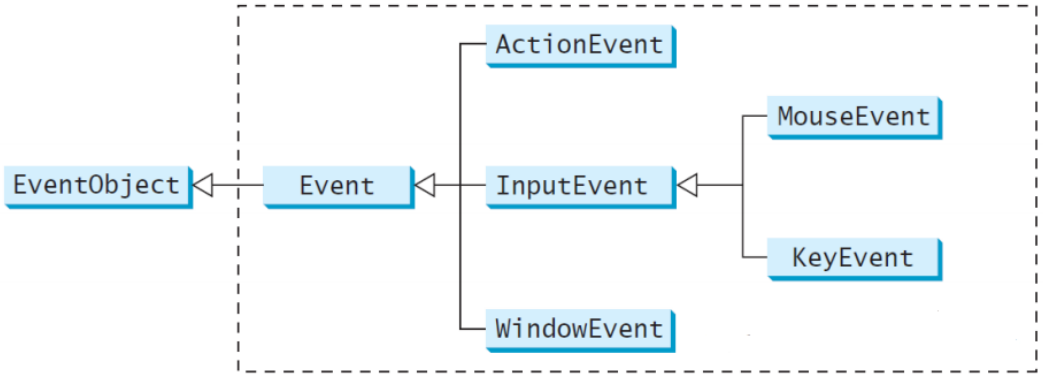
XỬ LÝ SỰ KIỆN TRONG JAVAFX

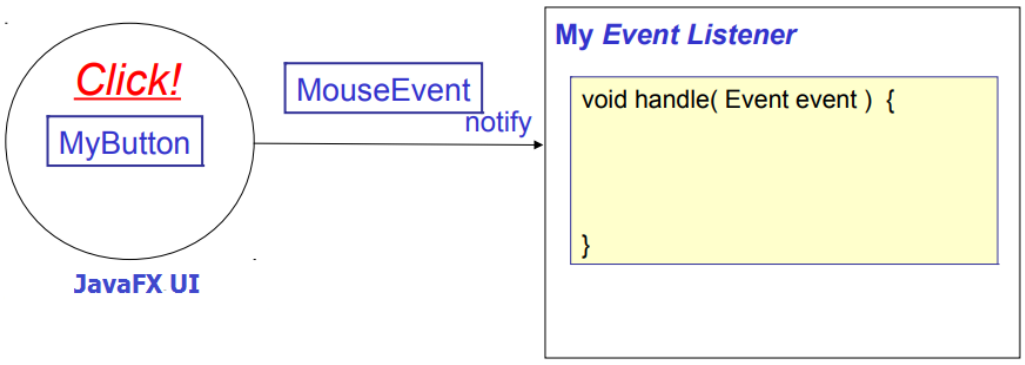
Trong JavaFX, chúng ta có thể phát triển ứng dụng GUI, ứng dụng web và ứng dụng đồ họa. Trong những ứng dụng đó, mỗi khi người sử dụng tương tác với các thành phần giao diện một sự kiện phát sinh.

Các hoạt động tương tác của người sử dụng có nhiều loại như click vào một nút nhấn, di chuột, nhập ký tự từ bàn phím, chọn một mục từ danh sách hay cuộn trang... Ứng với mỗi hoạt động đó là một loại sự kiện, mỗi thành phần giao diện khác nhau có thể phát sinh những sự kiện khác nhau. Tất cả các lớp sự kiện trong JavaFX đều thừa kế từ lớp nền tảng javafx.event.Event.



Cây thừa kế các lớp xử lý sự kiện trong JavaFX

Mô hình xử lý sự kiện



Mô hình xử lý sự kiện trong JavaFX

Quá trình một sự kiện diễn ra như sau:

+ Người sử dụng tương tác với một thành phần giao diện, ví dụ click chuột vào một nút nhấn.

+ Một sự kiện tương ứng được phát sinh, một đối tượng mô tả sự kiện được JavaFX tạo ra. Trong ví dụ này, JavaFX tạo ra một đối tượng MouseEvent mô tả những gì đã xảy ra, đóng gói trong đó thông tin về nguồn phát sinh sự kiện.

+ JavaFX tìm kiếm một trình xử lý sự kiện trong bộ lắng nghe sự kiện, nếu tìm thấy, phương thức handle trong trình xử lý sự kiện sẽ được gọi hoạt động với tham số đầu vào là đối tượng mô tả sự kiện đã tạo ra ở bước trước.

Các bước cài đặt xử lý sự kiện

- **Xác định nguồn tạo ra sự kiện và loại sự kiện**

+ Các sự kiện bắt nguồn từ sự tương tác với các thành phần giao diện (UI)

Bảng các sự kiện bắt nguồn từ sự tương tác với các thành phần giao diện (UI)

Tương tác của người dùng	Lớp sự kiện	Các thành phần UI hỗ trợ (Bao gồm cả các lớp con)
Các thao tác phím	KeyEvent	Node, Scene
Các thao tác chuột (trừ drag)	MouseEvent	Node, Scene
Thao tác drag chuột	MouseDragEvent	Node, Scene
Drag và Drop	DragEvent	Node, Scene
Cuộn trên thành phần UI	ScrollEvent	Node, Scene
Quay một thành phần UI	RotateEvent	Node, Scene
Yêu cầu hiển thị context menu	ContextMenuEvent	Node, Scene
Button được nhấn, Combobox được sổ (shown) hay ẩn (hidden) danh sách, Menu item được lựa chọn	ActionEvent	ButtonBase, ComboBoxBase, ContextMenu, MenuItem, TextField
Item trong một List, Table, Tree được chỉnh sửa	ListView.EditEvent TableColumn.CellEditEvent TreeView.EditEvent	ListView TableColumn TreeView
Trình Media sinh lỗi	MediaErrorEvent	MediaView
Menu được sổ (shown) hay ẩn (hidden)	Event	Menu
Popup được ẩn	Event	PopupWindow
Tab được lựa chọn hay đóng	Event	Tab
Window được đóng, hiển thị (shown) hay ẩn (hidden)	WindowEvent	Window

- **Đăng ký trình xử lý sự kiện với bộ lắng nghe**

+ Sử dụng phương thức tiện ích setOnXXX của chính thành phần giao diện

Các phương thức tiện ích để thêm sự kiện của thành phần giao diện có dạng:

setOnEvent-type(EventHandler<? super event-class> eventHandler)

Trong đó Event-type là loại sự kiện cụ thể ứng với lớp sự kiện (event-class). Ví dụ, ứng với lớp sự kiện MouseEvent ta có các loại sự kiện như MOUSE_CLICKED, MOUSE_MOVED. Khi đó phương thức tiện ích có dạng setOnMouseClicked, setOnMouseMoved.

Ví dụ: button.setOnMouseClicked(eventHandler);

+ Sử dụng phương thức addEventHandler của lớp cha javafx.scene.Node

addEventHandler(EventType<T> eventType,

EventHandler<? super T> eventHandler)

Ví dụ: button.addEventHandler(EventType.MOUSE_CLICKED, eventHandler);

+ Sử dụng phương thức addEventFilter của lớp cha javafx.scene.Node

addEventFilter(EventType<T> eventType,

EventHandler<? super T> eventFilter)

Ví dụ: button.addEventFilter(EventType.MOUSE_CLICKED, eventFilter);

Trình lọc sự kiện EventFilter luôn được thực hiện trước trình xử lý sự kiện EventHandler, nó thường được sử dụng cho các thành phần giao diện cha (nút cha) để cung cấp xử lý chung cho các nút con của nó hoặc chặn một sự kiện hay ngăn các nút con hoạt động trên sự kiện.

Ví dụ sau sẽ vô hiệu hóa bất kỳ sự kiện chuột nào đối với tất cả các thành phần con trong vùng chứa:

```
parentContainer.addEventFilter(
        MouseEvent.ANY,
        new EventHandler<MouseEvent>() {
            public void handle(final MouseEvent mouseEvent) {
                    mouseEvent.consume();
             }
        });
```

- **Cài đặt trình xử lý sự kiện Event Handler**

Trong JavaFX các trình xử lý sự kiện được cài đặt bằng cách thực thi một interface duy nhất EventHandler:

public interface EventHandler<T extends Event> extends EventListener{

void handle(T event)

}

Trong đó T là lớp sự kiện ta cần xác định ở bước trước hoặc lớp cha của lớp sự kiện ta xác định được. Ví dụ, khi click chuột vào một Button, ta có thể bắt sự kiện MouseEvent (có thể thay bằng InputEvent là lớp cha) hoặc ActionEvent. Khi đó trình xử lý sự kiện được cài đặt theo mẫu:

- Sử dụng inner class (lớp nội):

```
class Main{
        ...
        class ButtonHandler implements EventHandler<ActionEvent> {
               public void handle(ActionEvent evt) {
                      // mã cài đặt
               }
        }
        ...
```

```
        button.setOnAction(new ButtonHandler());
    }
```

- Sử dụng Anonymous Class. Cách này có thể sử dụng lại cho nhiều đối tượng được khuyến khích dùng.

```
class Main{
        ...
    EventHandler buttonHandler = new EventHandler<ActionEvent>(){
        public void handle(ActionEvent evt) {
            // mã cài đặt
        }
    };
    button.setOnAction(buttonHandler);
}
```

- Cài đặt nội dung ngay trong phương thức gọi

```
button.setOnAction(new EventHandler<ActionEvent>(){
    public void handle(ActionEvent evt) {
        // mã cài đặt
    }
});
```

- Sử dụng biểu thức Lambda (từ phiên bản Java 8)

```
class Main{
...
    EventHandler<ActionEvent>  buttonHandler = (event) -> {
        // không cần khai báo phương thức handle
        // chỉ cần mã cài đặt nội dung phương thức handle
    } ;
    button.setOnAction(buttonHandler);
}
```

- **Hủy bỏ một trình xử lý sự kiện**

Với mỗi Node ta có thể hủy bỏ một trình xử lý sự kiện đã đăng ký bằng phương thức:

removeEventHandler(EventType<T> eventType,
EventHandler<? super T> eventHandler)

removeEventFilter(EventType<T> eventType,
EventHandler<? super T> eventFilter)

Ví dụ: button. removeEventHandler(EventType.MOUSE_CLICKED, eventHandler)

Đối với các sự kiện được đăng ký bằng cách sử dụng phương thức tiện ích, ta có thể hủy bỏ đăng ký bằng cách truyền gọi lại phương thức tiện ích với tham số null.

Ví dụ: button.setOnAction(null);

Các sự kiện bắt nguồn từ sự thay đổi giá trị thuộc tính của UI

Nhiều thuộc tính của các đối tượng trong JavaFX có kiểu Property. Ví dụ, đối tượng javafx.scene.shape.Circle có các thuộc tính tọa độ (centerX, centerY), bán kính (radius) có kiểu DoubleProperty.

Các thuộc tính này có thể truy xuất vào với các phương thức getter, setter như JavaBean.

Ví dụ:

```
Circle circle = new Circle(5.5);
System.out.println("Circle radius = " + circle.getRadius());
          circle.setRadius(10.5);
 System.out.println("Circle radius = " + circle.getRadius());
```

Ngoài ra trong các thuộc tính Property còn nắm giữ các thông tin về đối tượng (metadata) như tên, giá trị và cả đối tượng bean.

Ví dụ:

```
      System.out.println(circle.radiusProperty());
```

Lệnh này sẽ in ra kết quả:

```
DoubleProperty [bean: Circle[centerX=0.0, centerY=0.0, radius=10.5,
            fill=0x000000ff], name: radius, value: 10.5]
```

Ta có thể lấy ra các thành phần chứa đựng trong một thuộc tính radius bằng các phương thức của đối tượng DoubleProperty:

```
circle.radiusProperty().getValue()
circle.radiusProperty().getName()
circle.radiusProperty().getBean()
circle.radiusProperty().get()
```

Một điểm khác biệt nữa mà ta muốn đề cập ở đây đó là, các thuộc tính kiểu Property của các đối tượng trong JavaFX là các thuộc tính có thể cài đặt lắng nghe sự kiện. Có hai loại sự kiện có thể cài đặt:

- InvalidationListeners: Cài đặt bắt sự kiện khi giá trị thuộc tính lẽ ra cần thay đổi rồi nhưng nó vẫn chưa được thay đổi
- ChangeListeners: Cài đặt bắt sự kiện khi giá trị thuộc tính thay đổi

Trong ví dụ về thành phần giao diện tương tác ChoiceBox ta đã cài đặt sự kiện ChangeListeners để bắt sự kiện thay đổi giá trị lựa chọn.

Đối với hình tròn ta có thể cài đặt sự kiện ChangeListeners để bắt sự kiện thay đổi giá trị bán kính bằng đoạn mã sau:

```
circle.radiusProperty().addListener(new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
                    Number oldValue, Number newValue) {
                System.out.println("New radius: " +
                                    newValue. doubleValue());
            }
});
```