Secteur Tertiaire Informatique

Filière « Etude et développement »

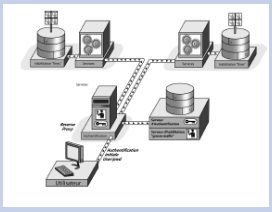
TP – Développer des interfaces graphiques

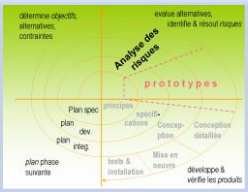
IHM Java FX - Formulaire

Apprentissage

Mise en situation

Evaluation





Introduction

A l'issue de cette activité, vous serez capable de mettre en place des fenêtres d'application en utilisant la bibliothèque JavaFX.

Ceci est la première étape vers la construction d'applications graphiques complexes en utilisant le paradigme de programmation événementiel.

La documentation officielle de JavaFX est disponible à l'adresse suivante : https://openjfx.io/

L'ensemble du fonctionnement de JavaFX est détaillé par la série d'articles suivantes : https://jenkov.com/tutorials/javafx/index.html

1. MISE EN PLACE DU PROJET

Afin de créer un projet JavaFX vierge basé sur Maven il vous est possible d'utiliser un archétype existant.

Par exemple, en utilisant la commande donnée par l'article suivant : https://openjfx.io/openjfx-docs/#maven

Attention

Comme souvent, l'archétype ne génère pas un fichier « pom.xml » avec les versions des biliothèques et des plugins les plus à jour.

Il vous faudra probablement changer :

- La version de Java que vous utilisez
- La version de la dépendance « **javafx.controls** ». Vous pourrez trouver les versions disponibles sur le site suivant : https://central.sonatype.com/artifact/org.openjfx/javafx-controls
- La version du plugin Maven d'aide à la compilation pour JavaFX : https://central.sonatype.com/artifact/org.openjfx/javafx-maven-plugin

Quand vous choisissez une version faites bien attention de ne pas sélectionner « ea » qui indique « early access ».

2. PREMIERE FENETRE

2.1 CREATION DE LA FENETRE

A faire

Suivez les étapes du tutoriel disponible sur le site suivant pour coder votre première fenêtre : https://jenkov.com/tutorials/javafx/your-first-javafx-application.html

2.2 UTILISATION D'UN PANE « VBOX »

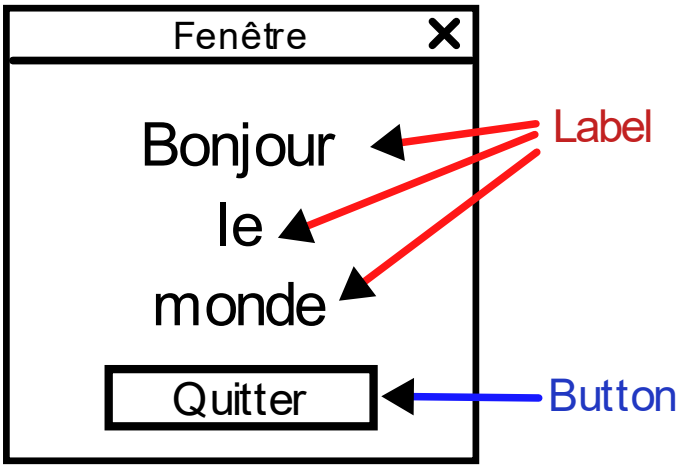
La construction d'interface graphique consiste souvent à organiser des « **conteneurs** » pour qu'ils puissent apparaître agencés comme prévu par la maquette.

Il est possible d'ajouter des composants graphiques dans les conteneurs qui seront affichés à l'écran.

Pour mieux comprendre l'utilisation de conteneurs vous allez intégrer le « wireframe » suivant :

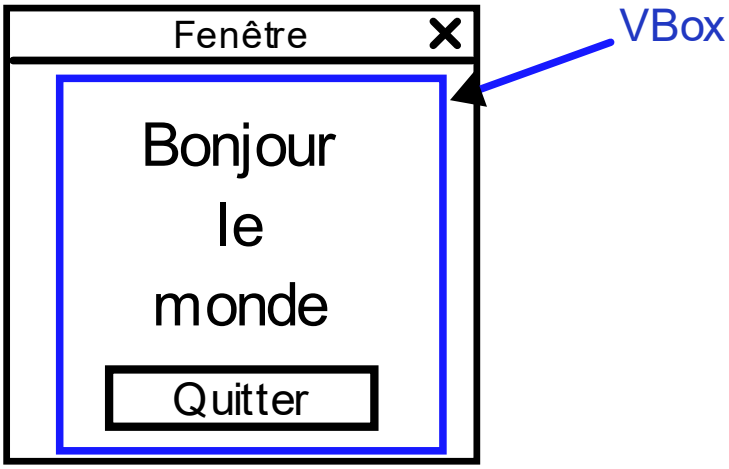
Vous allez devoir utiliser deux composants graphiques pour mener à bien le développement :

- Des « Label » (https://jenkov.com/tutorials/javafx/label.html)
- Un bouton (https://jenkov.com/tutorials/javafx/button.html)



Nous pouvons observer que ces composants sont organisés verticalement, nous pouvons donc utiliser une « VBox » :

- https://jenkov.com/tutorials/javafx/vbox.html



A faire

Suivez l'article à l'adresse https://jenkov.com/tutorials/javafx/vbox.html et implémentez le wireframe proposé.

2.3 GESTION DES EVENEMENTS

Vous allez maintenant ajouter l'action de fermeture de l'application lorsque l'utilisateur clique sur le bouton « **Quitter** ».

Pour se faire vous allez intégrer dans votre application une gestion d'évènements.

Information

Les informations qui suivent sont issues de la documentation Oracle :
https://docs.oracle.com/javafx/2/events/processing.htm

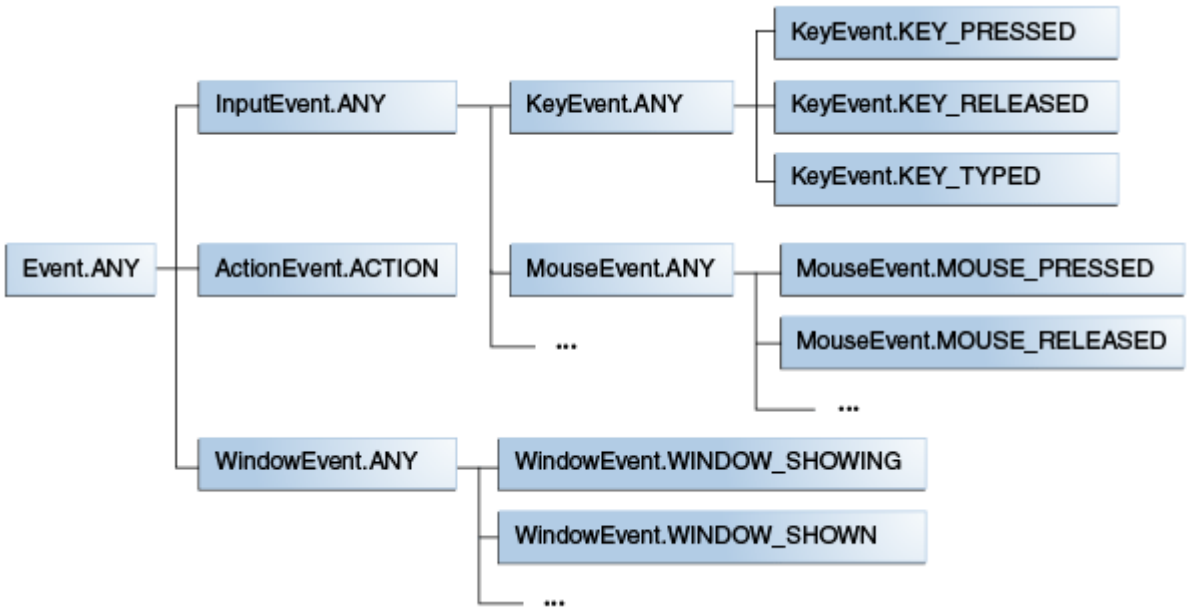
Dans les applications JavaFX, les événements signifient que quelque chose s'est produit, par exemple : l'utilisateur a cliqué sur un bouton, appuie sur une touche, déplace une souris ou effectue
d'autres actions.

Dans JavaFX, un événement est une instance de la classe « **javafx.event.Event** » ou de toute
sous-classe d'Event.

JavaFX fournit plusieurs sous-classe d'Event, notamment DragEvent, KeyEvent, MouseEvent, ScrollEvent et d'autres.

Vous pouvez définir votre propre sous-classe d'Event en étendant la classe Event.
Chaque événement inclut les informations suivantes : **Event Type**, **Source** et **Target**.

Ci-dessous, la hiérarchie des évènements :



Le type d'événement de niveau supérieur dans la hiérarchie est **Event.ROOT**, qui est équivalent à **Event.ANY**.

Dans les sous-types, le type d'événement ANY est utilisé pour désigner n'importe quel type d'événement dans la classe d'événements. Par exemple, pour fournir la même réponse à n'importe quel type d'événement de type « touche », utilisez KeyEvent.ANY comme type d'événement pour le filtre d'événements ou le gestionnaire d'événements.

Les gestionnaires d'événements (**EventHandler**) vous permettent de gérer les événements.
Un nœud peut avoir un ou plusieurs gestionnaires pour gérer un événement.
Un seul gestionnaire peut être utilisé pour plusieurs nœuds et plusieurs types d'événements.

A faire

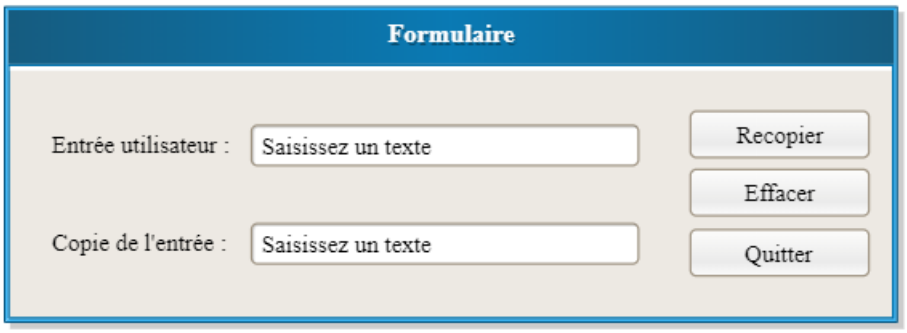
Inspirez-vous de l'article disponible à l'adresse https://www.baeldung.com/javafx-button-eventhandler pour implémenter le clic sur le bouton.

La fonction qui se déclenche lors de cet évènement devra fermer l'application. Voici une méthode utilisable pour fermer l'application : « Platform.exit() ».

3. PREMIER FORMULAIRE

3.1 OBJECTIF DE DEVELOPPEMENT

Vous allez modifier le code de votre application pour mettre en place le formulaire comme détaillé par le modèle suivant :

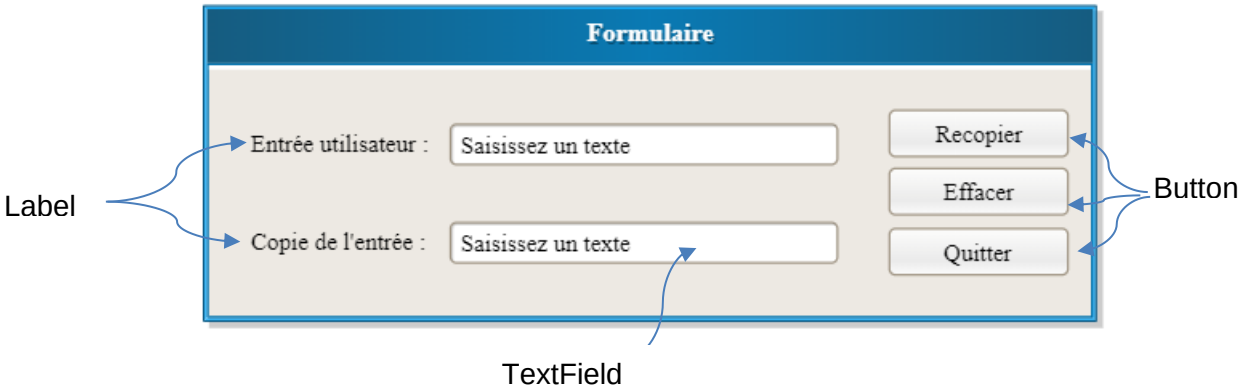


Les fonctionnalités à implémenter sont les suivantes :

- En cliquant sur « Recopier » la saisie utilisateur devra être copiée dans le champ de texte associé à « Copie de l'entrée » ;
- En cliquant sur « Effacer » les deux champs de texte doivent être effacés ;
- En cliquant sur « Quitter » l'application doit se terminer (méthode pour fermer une application : « Platform.exit() ») ;
- Le champ de texte associé à « Copie de l'entrée » **ne doit pas pouvoir être modifiable et doit être grisé** (méthodes de la classe TextField à utiliser : « setEditable » et « setDisable »).

Il y a trois composants graphiques à utiliser :

- Des « Label » (https://jenkov.com/tutorials/javafx/label.html)
- Des « TextField » (https://devstory.net/11093/javafx-textfield)
- Des « Button » (https://jenkov.com/tutorials/javafx/button.html)



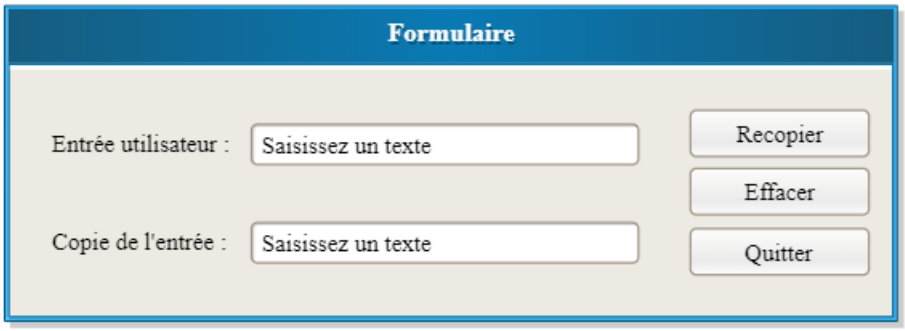
3.2 INTEGRATION DES COMPOSANTS GRAPHIQUES

Vous allez utiliser le système de « pane » afin d'organiser les différents composants graphiques de la scène.

Pour l'organisation de vos composants du formulaire vous pourrez utiliser un « GridPane » pour disposer les éléments dans une grille ainsi qu'un « **VBox** »

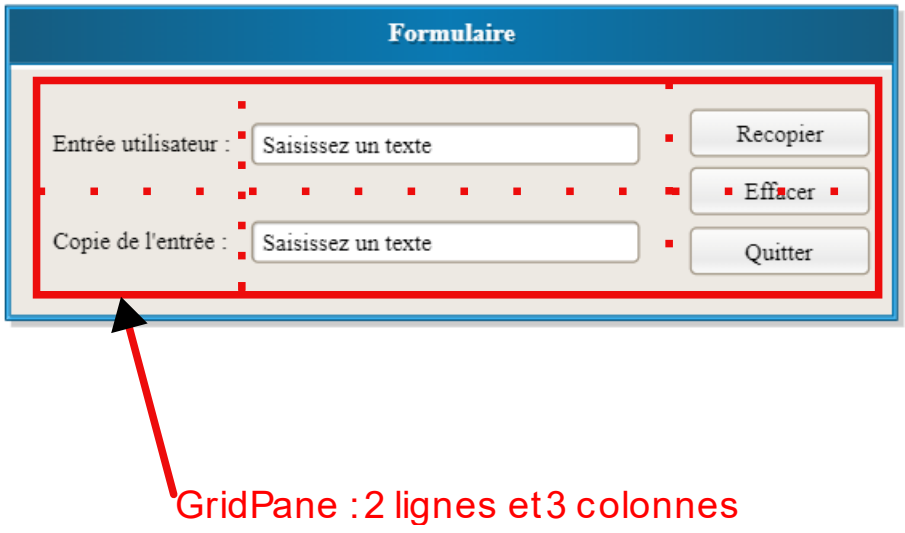
Lisez l'article suivant pour mieux comprendre ce qu'est un « grid pane » et un Vbox :

- https://jenkov.com/tutorials/javafx/gridpane.html
- https://jenkov.com/tutorials/javafx/vbox.html



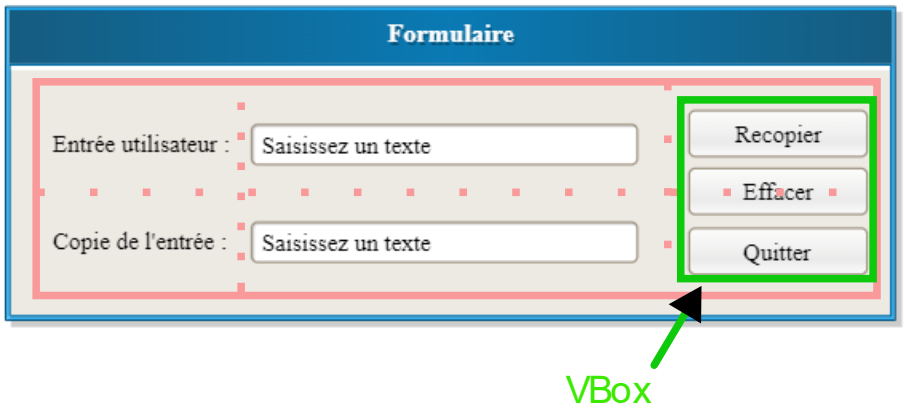
Pour un descriptif des différents « Pane » utilisable vous pouvez vous référer à cette ressource : https://www.tutorialspoint.com/javafx/javafx_layout_panes.htm.

Voici, par exemple un découpage visuel de la construction du « GridPane » que vous pourrez utiliser.



Vous remarquerez que les buttons à droite sont regroupés et s'étendent sur 2 lignes.

Pour se faire vous pourrez utiliser une « VBox » que vous ajouterez au « GridPane », comme présenté par la figure ci-dessous :



3.3 IMPLEMENTATION DES ACTIONS SUR LES BOUTONS

3 boutons sont disponibles :

- « **Recopier** » va prendre le texte du « TextField » « Entrée utilisateur » et le recopier dans « Copie de l'entrée »
- « **Effacer** » va effacer le contenu de « Copie de l'entrée »
- « **Quitter** » va quitter l'application

Ajoutez 3 gestionnaires d'évènements pour implémenter ces actions.

Pour modifier le texte contenu dans un « TextField » vous pourrez utiliser la méthode « setText » (documentation disponible ici : https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextInputControl.html#setText-java.lang.String-).

A faire

Ajoutez la gestion des évènements qui se déclencheront sur les clics des différents boutons et implémentez les actions.

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Michel Coulard – Formateur Evry
Chantal Perrachon – IF Neuilly sur Marne

Date de mise à jour : 08/07/2024