JAVA

TUTORIEL JavaFX et l'IHM

Ce tutoriel va vous donner les bases pour créer une IHM avec JavaFX. Anciennement, on utilisait SWING, pour lequel vous pourrez trouver de nombreux tutoriels sur Internet. Il est plus courant et standard d'utiliser JavaFX aujourd'hui.
On peut créer des interfaces entièrement en Java, comme dans les deux premiers exemples, mais il est plus simple et plus efficace de créer une interface générée en FXML, dérivée du XML et associée au CSS, d'autant que le logiciel Scene Builder permet de le faire graphiquement.
De plus, cela permet de créer en plus ou en même temps des interfaces pour des applications mobiles ou cloud (voir le site de Gluon ci-après).
Le troisième exemple, un carnet d'adresses à partir d'un tutoriel de *Marco Jakob*, en est l'illustration.

A. Installation

- Les fichiers à télécharger sont déjà avec ce tutoriel dans les fichiers. Vous prendrez soin de prendre les mêmes versions de java, javafx et scene builder.

- Installation de e(fx)clipse :
  - Allez dans le menu « Help / Market Place ». Rechercher et installer « e(fx)clipse ». Redémarrer
  - Allez dans le menu « Help / Install New Software », Dans « Work with » coller l'adresse suivante : `https://download.eclipse.org/efxclipse/updates-nightly/site/` cliquez sur « Add ». Cocher les 2 cases qui apparaissent dans la liste en dessous. Cliquer sur « Next », « Finish » et « Restart ». Cela permet de mettre à jour e(fx)clipse en version 3.10.

- **Pour windows :**
  - Télécharger openjfx sdk (https://gluonhq.com/products/javafx/) et le décompresser dans un répertoire sans caractères spéciaux en enlevant les répertoires de version (*C:\Divers\javafx* par exemple et pas *C:\Divers\javafx\openjfx-18.0.2_linux-x64_bin-sdk\javafx-sdk-18.0.2 pour éviter les problèmes de version par la suite*)
  - Télécharger et installer Scene Builder (mettez-le aussi dans un répertoire type *C:\Divers\SceneBuilder)* : https://gluonhq.com/products/scene-builder/
  - Dans Eclipse, dans le menu *Window/Preferences/JavaFX*, mettre :
    - le chemin de l'exécutable scenebuilder : *C:\Divers\SceneBuilder\SceneBuilder.exe*
    - le path de javafx : *C:\Divers\javafx* (par exemple, vous adaptez selon votre répertoire)

- **Pour Linux Ubuntu :**
  - Désinstaller openjfx : *sudo apt remove openjfx*, puis nettoyer avec *sudo ./maj.bash*
  - Télécharger javafx sdk (https://gluonhq.com/products/javafx/) et le décompresser dans un répertoire sans caractères spéciaux en enlevant les répertoires de version (*/home/cy/javafx* par exemple et pas */home/cy/javafx/openjfx-18.0.2_linux-x64_bin-sdk\javafx-sdk-18.0.2*).
    À la fin du fichier *.bashrc*, rajouter la ligne ci-dessous, puis redémarrez :
    export PATH_TO_FX="/home/cy/javafx/lib"
  - Télécharger Scene Builder : https://gluonhq.com/products/scene-builder/
    et l'installer : *sudo apt install -f ./SceneBuilder-18.0.0.deb* (ou la version en cours)
  - Dans Eclipse dans le menu *Window/Preferences/JavaFX*, mettre :
    - le path de l'exécutable scenebuilder : */opt/scenebuilder/bin/SceneBuilder*
    - le path de javafx : */home/cy/javafx/lib* (à modifier pour vous)

- **Créer la User Library javafx :**
  - Aller dans le menu Eclipse/Preferences/Java/Build Path/User Libraries/
  - Créer la librairie : New "javafx" => Add External JARS =>
    - Windows : *C:\Divers\javafx\lib* et sélectionner tous les jar de javafx et les inclure
    - Linux : */home/cy/javafx/lib* et sélectionner tous les jar de javafx et les inclure
  - Apply and close
- Cette vidéo peut vous aider visuellement à réaliser cette installation : https://www.youtube.com/watch?v=bk28ytggz7E&feature=emb_rel_end

B. Projet de test en Java pur HelloWorldJFX :

- On va créer un projet avec une fenêtre vide et une fenêtre lançant un « Hello World ».
- Nouveau Projet/JavaFX/JavaFX Project : *HelloWorldJFX*
- Décocher la création du module-info.java (sinon le supprimer après)
- Click Next => Libraries, Click sur Classpath, puis "Add Library" => "JavaFX SDK" et recommencer pour "User Library" => Next => cocher javafx => Finish => Finish
- Supprimer module-info java
- Ouvrir Main.java et cliquer dans le code
- Aller dans le Run Configuration/Java Application/main, onglet Arguments/VM arguments, rajouter :
  - Windows : *--module-path "C:\Divers\javafx\lib" --add-modules javafx.controls,javafx.fxml*
  - Linux : *--module-path "/home/cy/javafx/lib" --add-modules javafx.controls,javafx.fxml*
- Exécuter avec "Run" => cela lance une fenêtre vide
- Dans le package *application*, Créer la classe "*HelloWorldJFX*" et remplacer le code par celui ci-dessous :

```
package application;

import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloWorldJFX extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World!");

        // Definition d'un bouton avec action sur clic
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane();
        root.getChildren().add(btn);
        primaryStage.setScene(new Scene(root, 300, 250));
        primaryStage.show();
    }
}
```

- Inclure à nouveau le *module-path* vu au-dessus dans le *Run Configuration* et exécuter pour valider l'application. En cliquant sur le bouton, vous voyez apparaître « Hello World » dans la fenêtre d'exécution d'Eclipse. Lire le code pour comprendre les actions.

C. Projet de test graphique en Java pur TestGraphicJFX :

- Toujours dans le même projet (sinon vous devez recréer un projet en suivant les étapes vues précédemment), créer la classe *TestGraphicJFX* et inclure le code suivant :
  Pour rappel, vous ferez attention aux erreurs de recopie (guillemets et autres)

```
package application;

import javafx.animation.FadeTransition;
import javafx.animation.TranslateTransition;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;
import javafx.util.Duration;

public class TestGraphicJFX extends Application {

   /* objet graphique représentant un rectangle */
   public Rectangle rectangle;

   /* definir la troupe des objets graphiques */
   Group root;

   /* lancement de l'application */
   public void start(Stage primaryStage) {

        construireScene(primaryStage);
   }

   /* construction des objets affichés */
   void construireScene(Stage primaryStage) {
        int largeur = 800;
        int hauteur = 600;
        // definir la troupe
        root = new Group();
        // definir la scene principale
        Scene scene = new Scene(root, largeur, hauteur, Color.BLACK);
        primaryStage.setTitle("Test de fenêtre...");
        primaryStage.setScene(scene);

        // definir les objets graphiques, ici un rectangle
        rectangle = new Rectangle(largeur - largeur / 10, hauteur / 2, largeur / 10,
hauteur / 10);
        rectangle.setFill(Color.CHARTREUSE);
        // ajouter une gestion d'événement clic souris au rectangle
        rectangle.setOnMouseClicked(event -> {
              System.out.println("mouse click detected! " + event.getSource());
              System.out.println("click on  " + event.getX() + "," + event.getY());
        });
        // ajouter le rectangle à la troupe
        root.getChildren().add(rectangle);

        // une transition sur l'opacité de l'objet
        FadeTransition ft = new FadeTransition(Duration.millis(2000), rectangle);
```

```
        ft.setFromValue(1.0);
        ft.setToValue(0.01);
        ft.setCycleCount(2);
        ft.setAutoReverse(true);
        ft.play();

        // une transition sur la position de l'objet
        TranslateTransition tt = new TranslateTransition(Duration.millis(3000),
rectangle);
        tt.setFromX(0);
        // coordonnée relative
        tt.setToX(-largeur + largeur / 10);
        tt.setCycleCount(1);
        tt.play();

        // afficher le theatre
        primaryStage.show();
   }

   /* lancement du prog */
   public static void main(String[] args) {
        launch(args);
   }
}
```

- Inclure à nouveau le *module-path* vu au-dessus dans le Run Configuration et exécuter pour valider l'application. Cliquer dans le rectangle et constatez le résultat dans la fenêtre d'exécution d'eclipse. Lire le code pour comprendre les actions.
- On ne s'attardera pas sur les possibilités de créer en Java une interface complète ou en générer de nouveaux éléments à tout moment, mais vous savez désormais que c'est possible.

D. Création d'une application avec Scene Builder, FXML et CSS

- Afin de créer une application complète de carnet d'adresse, suivez le tutoriel : https://code.makery.ch/fr/library/javafx-tutorial/
- Cela vous permettra de maîtriser la création de l'interface en FXML / CSS / Java avec Scene Builder, et créer une application complète jusqu'à son exécutable.
- Vous trouverez à la fin de chaque chapitre des articles intéressants qui pourront vous aider à aller plus loin et réaliser votre projet plus facilement.
- Présents dans les corrections fournies, vous trouverez les icônes de l'application dans le répertoire de même nom dans les fichiers Teams.
- **Attention :** Pour la Partie 5, stockage des données en XML, JAXB n'étant plus inclus dans les JDK ≥ 11, il faut les inclure manuellement dans le projet. Vous les trouverez dans le répertoire JAXB dans les fichiers Teams.
  Pour cela, créer un répertoire *lib* dans le répertoire *resource* (là où vous avez mis les icônes), par clic droit sur *resource*, *New/Folder*. Ensuite, glisser-déposer les jar dans ce répertoire. Puis, dans le Build Path/Libraries, faire un clic droit sur *Classpath* et cliquer sur « Add JARs ». Ajouter alors les JAR déposés dans *lib*, puis *Apply* et *Apply and Close*, comme pour javafx.
  Dès lors, dans le tutoriel, lors des imports de jaxb, remplacer « *import jaxb…* » par « *import javax…* ».
  Cela étant, vous remarquerez qu'il est possible de faire le même genre de sauvegarde en utilisant la sérialisation ou la sauvegarde en JSON ou CSV, vue dans le cours sur les entrées-sorties.