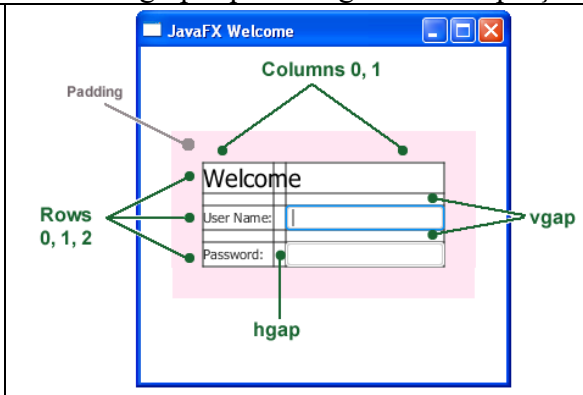
TP5 : IHM -JavaFX Simple

Lors de création d'une application JavaFx sous Eclipse IDE, on procède de la même manière. Seulement, au lieu de choisir pour « New Project » la catégorie « Java → Java Application » ça sera « JavaFX → JavaFX Application »...

Exercice 1 : création d'une interface graphique Login

L'objectif est de créer une interface graphique de login selon l'aperçu suivant :



conception

réalisation

1) Compléter le code de l'application (on vous donne le code de la partie événementielle) :

```
public class Login extends Application {
    @Override
   public void start(Stage primaryStage) {
   primaryStage.setTitle("JavaFX Welcome");
      GridPane grid = new GridPane();
// à compléter
final Text actiontarget = new Text();
      grid.add(actiontarget, 1, 6);
      btn.setOnAction(new EventHandler<ActionEvent>() {
         @Override
         public void handle(ActionEvent e) {
            actiontarget.setFill(Color.FIREBRICK);
            actiontarget.setText("Sign in button pressed"); } });
// à completer
}
public static void main(String[] args) {
      launch(args);    }}
```

2) Tester le changement de style de l'interface avec une feuille CSS :

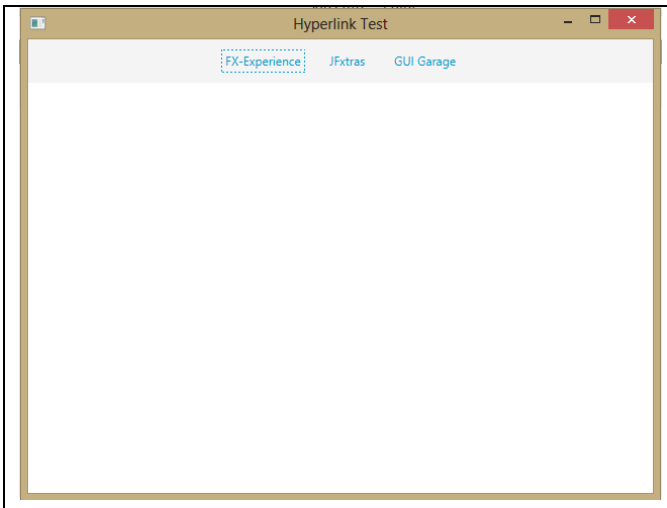
Vous aurez à :

- ✓ mettre l'image : Background.jpg dans le répertoire du projet login → src
- ✓ créer un fichier Login.css sous Eclipse pour le même projet :
  New File → Other → Cascading Style Sheet ……
- ✓ ajouter le code fourni du style
- ✓ ajouter dans la méthode start() de l'application JavaFx cette ligne :

```
scene.getStylesheets().add(Login.class.getResource("login.css").toExternalForm());
```

Exercice 2 : création d'une interface graphique avec hyperliens

L'objectif est créer une interface graphique de liaisons avec des sites internet selon l'aperçu suivant :

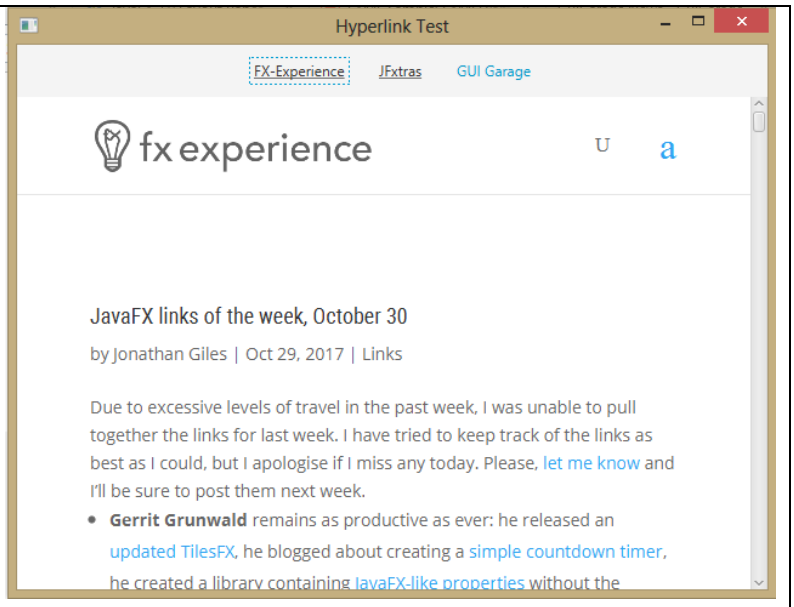


Trois liens vers les sites :

http://fxexperience.com

http://jfxtras.org

http://www.guigarage.com



Loading de : http://fxexperience.com

Pour réaliser cette interface, compléter le code suivant de l'application (on vous donne le code de la partie événementielle) :

```
public class Hper extends Application {
    private String[] tCaption = {"FX-Experience", "JFxtras","GUI Garage"};
    private String[] tUrl = {"http://fxexperience.com", "http://jfxtras.org","http://www.guigarage.com"};
    private Hyperlink[] tLinks = new Hyperlink[tUrl.length];
// compléter
@Override
    public void start(Stage primaryStage) {
// completer
/*on peut utiliser le composant WebView qui contient un moteur de rendu de pages web de type WebEngine (basé sur le projet open-source WebKit)*/
    WebView browser = new WebView();
```

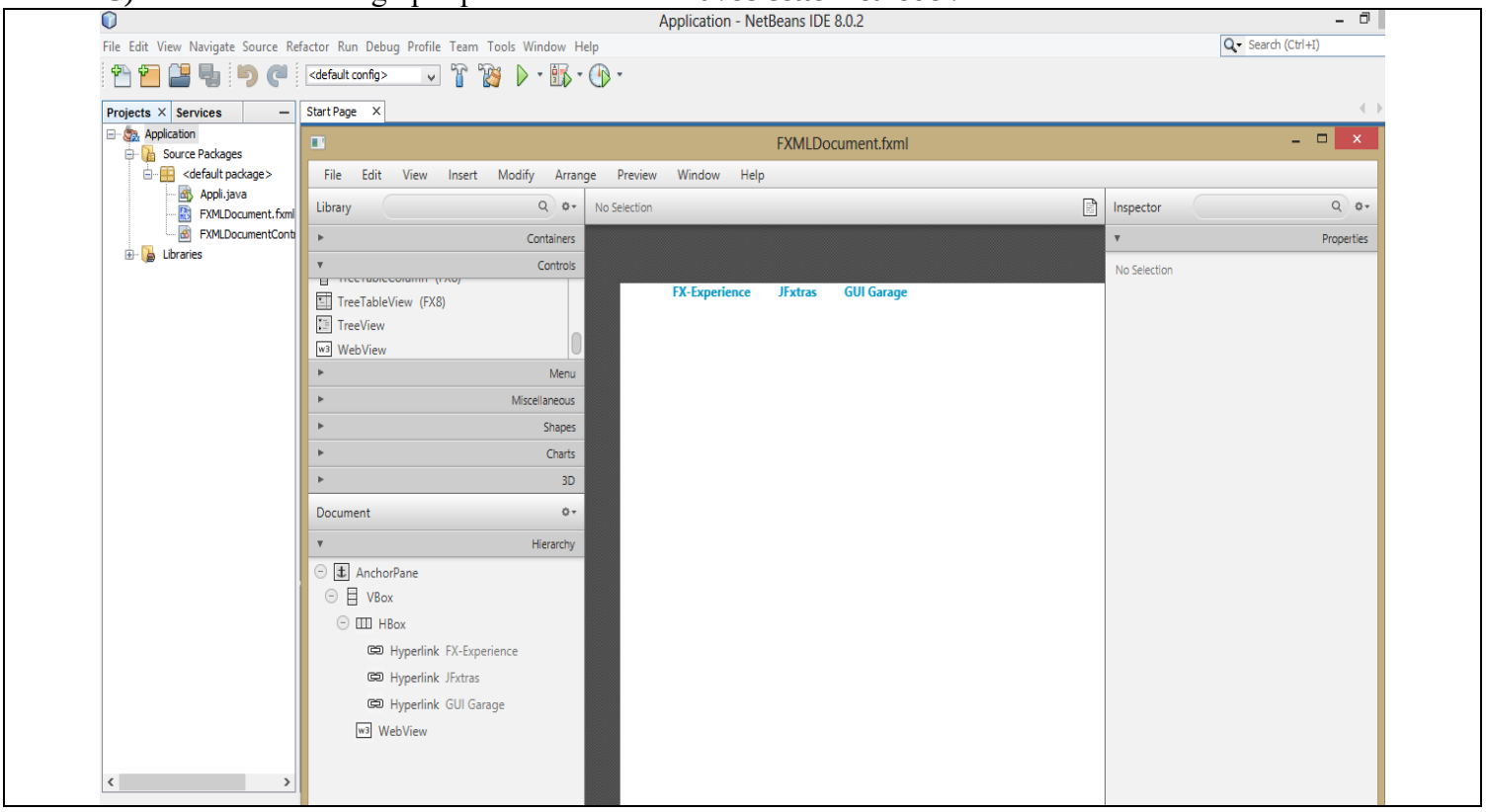
```
    WebEngine wEngine = browser.getEngine();
    for (int i=0; i<tUrl.length; i++) {
// completer
// partie événémentielle
tLinks[i].setOnAction(event -> { wEngine.load(url);});}
// compléter
```

Exercice 3 : Utilisation de Scene Builder 2

La méthode déclarative est la deuxième manière de création des interfaces graphiques en Java. Elle consiste à utiliser un outil interactif appelé « Scene Builder » qui génère un code FXML ouvrable par Eclipse IDE.

1) Téléchager et installer Scene Builder 2
2) Après la création d'un projet JavaFx, créer un nouveau fichier dans ce projet New File → JavaFX → Empty FXML (...). Découvrir les parties du fichier crée.

3) Ouvrir Scene Builder par clic sur le bouton droite au niveau du titre du fichier crée → open
4) Découvrir les différents éléments de cet outil. Voir aussi annexe du TP.
5) Créer l'interface graphique de l'exercice 2 avec cette méthode :



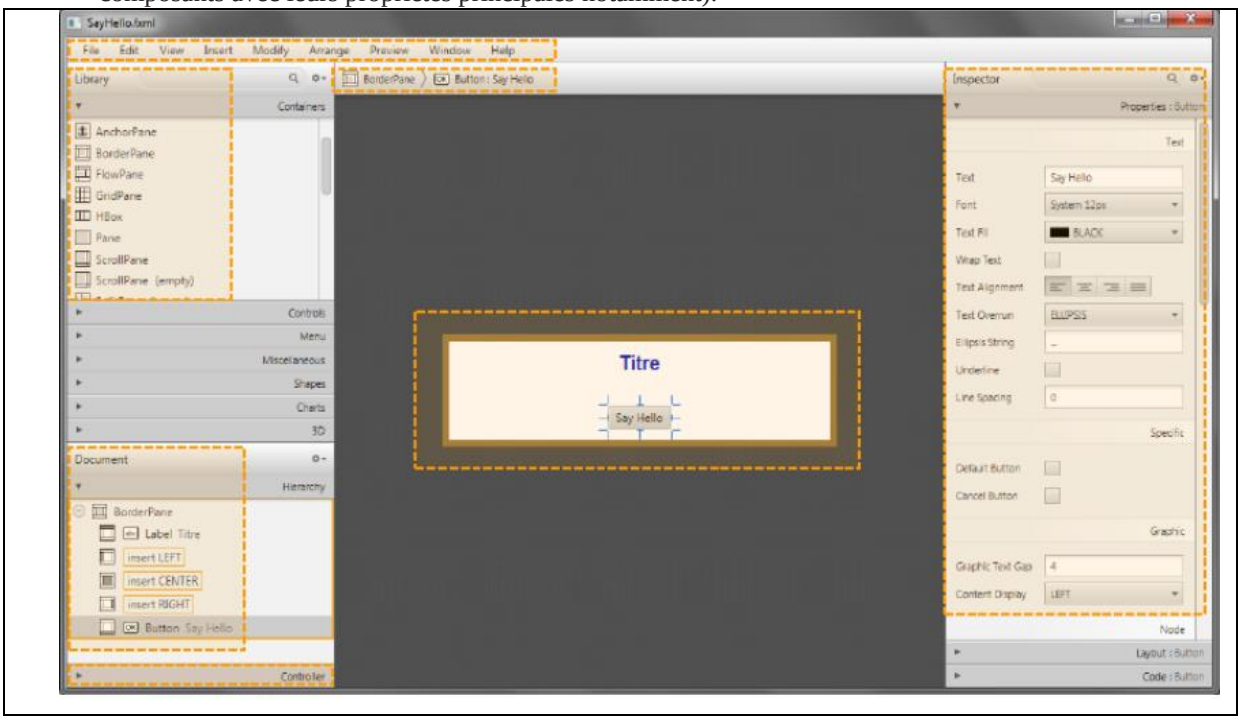
Annexe TP1 : FXML Scene Builder

Fichiers FXML

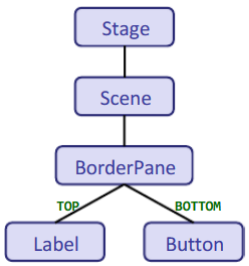
- Un fichier FXML est un fichier au format XML dont la syntaxe est conçue pour décrire l'interface avec ses composants, ses conteneurs, sa disposition, … Il décrit le « quoi » mais pas le « comment ».
- A l'exécution, le fichier FXML sera chargé par l'application (classe FXMLLoader) et un objet Java sera créé (généralement la racine est un conteneur) avec les éléments que le fichier décrit (les composants, conteneurs, graphiques, …).
- Il est possible de créer les fichiers FXML avec un éditeur de texte mais, plus généralement, on utilise un outil graphique (SceneBuilder) qui permet de concevoir l'interface de manière conviviale et de générer automatiquement le fichier FXML correspondant.
- Le langage FXML n'est pas associé à un schéma XML mais la structure de sa syntaxe correspond à celle des API JavaFX :
  - ✓ Les classes JavaFX (conteneurs, composants) peuvent être utilisées comme éléments dans la syntaxe XML
  - ✓ Les propriétés des composants correspondent à leurs attributs

Scene Builder

- L'outil graphique Scene Builder permet de concevoir l'interface de manière interactive en assemblant les conteneurs et les composants et en définissant leurs propriétés.
- Le mode de fonctionnement de cet utilitaire est assez classique avec une zone d'édition centrale, entourée d'un certain nombre d'outils : palettes de conteneurs, de composants, de menus, de graphiques, vue de la structure hiérarchique de l'interface, inspecteurs de propriétés, de layout , etc.
- L'utilisation de cet outil n'est pas décrite en détail dans ce support, il faut se référer à la documentation disponible. Son utilisation est cependant assez intuitive, pour autant que les éléments affichés soient connus (conteneurs, composants avec leurs propriétés principales notamment).



Exemple : Say Hello



- Variante procédurale : (idée générale)

  BorderPane root = new BorderPane();

```
root.setPrefWidth(250);
root.setPrefHeight(80);
root.setStyle("-fx-background-color: #FFFCAA");
root.setPadding(new Insets(10, 5, 10, 5));
Label lblTitle = new Label();
lblTitle.setText("Titre");
lblTitle.setTextFill(Color.web("#0022CC"));
lblTitle.setFont(Font.font("SansSerif", FontWeight.BOLD, 20));
BorderPane.setAlignment(lblTitle, Pos.CENTER);
root.setTop(lblTitle);
Button btnSayHello = new Button();
 btnSayHello.setText("Say Hello");
BorderPane.setAlignment(btnSayHello, Pos.CENTER);
root.setBottom(btnSayHello);
btnSayHello.setOnAction(event -> { lblTitle.setText("H e l l o !"); lblTitle.setTextFill(Color.FUCHSIA); });
```

- Variante déclarative :

Le fichier XML comporte tout d'abord une partie déclarative (en-tête et importations nécessaires à la création dynamique de l'objet Java).Ensuite on trouve la description de la structure de l'interface (graphe de scène avec conteneurs, composants et propriétés)

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>

<BorderPane . . . >

   . . .

</BorderPane>
```

Racine du graphe de scène.

Conteneurs et composants du graphe de scène.

```
<BorderPane prefHeight="80.0" prefWidth="250.0"
            style="-fx-background-color: #FFFCAA;"
            xmlns=http://javafx.com/javafx/8 xmlns:fx=http://javafx.com/fxml/1
            fx:controller="supp_cours.chap07.SayHelloController">
  <top>
    <Label id="title" fx:id="title" text="Titre" textFill="#0022cc"
           BorderPane.alignment="CENTER">
      <font>
        <Font name="SansSerif Bold" size="20.0" />
      </font>
    </Label>
  </top>
  <bottom>
    <Button fx:id="btnHello" onAction="#handleButtonAction" text="Say Hello"
            BorderPane.alignment="CENTER" />
  </bottom>
  <padding>
    <Insets bottom="10.0" left="5.0" right="5.0" top="10.0" />
  </padding>
</BorderPane>
```

La méthode start() de la classe principale peut charger le fichier FXML en invoquant la méthode statique FXMLLoader.load(url) qui prend en paramètre l'URL de la ressource à charger. La méthode getResource (name) de la classe Class permet de trouver (par le classloader) l'URL d'une ressource à partir de son nom :

```
public void start(Stage primaryStage) throws Exception {
//--- Chargement du fichier FXML
  BorderPane root = FXMLLoader.load (getClass().getResource("SayHello.fxml"));
Scene scene = new Scene(root);
primaryStage.setScene(scene);
primaryStage.setTitle("SayHello FXML");
primaryStage.show();}
```

Interprétation des fichiers FXML

Lors du chargement du fichier FXML, son contenu est interprété et des objets Java correspondants sont créés.
Par exemple, l'élément : <BorderPane prefHeight="80.0" prefWidth="250.0" . . .
sera interprété comme:
BorderPane rootPane = new BorderPane();
rootPane.setPrefHeight(80.0);
rootPane.setPrefWidth(250.0);. . .
Quand un attribut commence par le nom d'une classe suivi d'un point et d'un identificateur, par exemple : <TexField GridPane.columnIndex="3" . . .
L'attribut sera interprété comme une invocation de méthode statique
TextField tfd = new TextField();
GridPane.setColumnIndex(tfd, 3);

Pour les propriétés qui ne peuvent pas facilement être représentées par une chaîne de caractères, un élément est imbriqué (plutôt que de déclarer des attributs). Par exemple, si l'on considère l'élément :
<Label id="title" fx:id="title" text="Titre" textFill="#0022cc" BorderPane.alignment="CENTER"> <Font name="SansSerif Bold" size="20.0" /> </Label>
On constate que la propriété Font est codée comme un élément imbriqué dans l'élément Label
.Pour les propriétés de type liste (par exemple children), les éléments de la liste sont simplement imbriqués et répétés dans l'élément représentant la liste (par exemple, les composants enfants seront listés entre les balises <children> et </children>).