

Développement d'applications avec IHM (JavaFX)

IHM déclaratives avec FXML

Petru Valicov

`petru.valicov@umontpellier.fr`

`https://gitlabinfo.iutmontp.univ-montp2.fr/ihm`

2022-2023

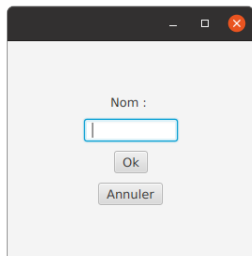


Problématique

```
public void start(Stage primaryStage) {
    VBox root = new VBox();
    Label label = new Label("Nom : ");
    TextField champ = new TextField();
    Button ok = new Button("Ok");
    ok.setOnAction(actionEvent -> {
        // du code pour traiter le clic
    });
    Button annuler = new Button("Annuler");
    annuler.setOnAction(actionEvent -> {
        // du code pour traiter le clic
    });

    // ici du code ajoutant pleins d'autres d'éléments à la scène
    // avec des handlers et des traitements

    root.getChildren().addAll(label, champ, ok, annuler);
    Scene scene = new Scene(root, 300, 300);
    primaryStage.setScene(scene);
    primaryStage.show();
}
```



Il est recommandé de séparer le "design" de l'application (structuration, composants graphiques) du code modélisant le comportement → **principe de responsabilité unique.**

Description des IHM - FXML

- Le FXML est un langage basé sur XML pour construire des interfaces graphiques JavaFX.
- Un fichier *.fxml* décrit la structuration du graphe de scène, les propriétés graphiques de chaque composant (taille, police, couleur etc.)
- Le code FXML est modifiable indépendamment du code métier de l'application
- Notamment, il est possible d'utiliser des outils interactifs de création d'interfaces graphiques (SceneBuilder à voir en TP)

Avantage principal

Une séparation nette entre le code métier et l'interface graphique utilisateur.

FXML - exemple

Fichier *hello-view.fxml* décrivant une fenêtre

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

<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

<?import javafx.scene.control.Button?>
<VBox alignment="CENTER" spacing="20.0"
  xmlns:fx="http://javafx.com/fxml">

  <Label text="Mon etiquette"/>
  <Button text="Hello!"/>
</VBox>
```



```
public class SalutLeMondeAvecFxml extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        FXMLLoader fxmlloader = new FXMLLoader(
            SalutLeMondeAvecFxml.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlloader.load(), 320, 240);
        stage.setTitle("Fenêtre construite avec FXML");
        stage.setScene(scene);
        stage.show();
    }
}
```

FXML - exemple

Fichier *hello-view.fxml* :

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

<?import javafx.scene.text.Font?>
<VBox alignment="CENTER" spacing="20.0"
  xmlns:fx="http://javafx.com/fxml/1"
  xmlns="http://javafx.com/javafx/17">

  <Label text="Mon etiquette"
    textFill="#cc3026">
    <font>
      <Font name="Arial" size="17.0"/>
    </font>
  </Label>
  <Button text="Hello!"/>
</VBox>
```

Code en procédural :

```
VBox root = new VBox();
root.setAlignment(Pos.CENTER);
root.setSpacing(20);
Label label = new Label("Mont etiquette");
label.setFont(Font.font("Arial", 17));
label.setTextFill(Paint.valueOf("#cc3026"));
Button bouton = new Button("Hello");
root.getChildren().addAll(label, bouton);
```

Comment ça marche ?

Le fichier *.fxml* décrit la structure de l'interface graphique.

- correspond au nœud racine du graphe de scène JavaFX
- pour chaque élément déclaré dans le *.fxml* (labels, boutons, conteneurs etc.), des objets Node correspondants sont instanciés par l'environnement JavaFX

Le chargement du *.fxml* a lieu dans la méthode `start()` :

```
/* l'objet de type FXMLLoader prend en paramètre du constructeur
   le chemin d'accès vers le fichier fxml */
FXMLLoader fxmlLoader = new FXMLLoader(
    SalutLeMondeAvecFXML.class.getResource("hello-view.fxml"));

// La méthode load() instancie l'objet Node décrit dans le fxml
Scene scene = new Scene(fxmlLoader.load(), 320, 240);
```

Une fois chargés, les Node décrits dans le *.fxml* sont équivalents à ceux pouvant être créés de manière procédurale.

FXML - exemple (avec actions et contrôleur)

Fichier *hello-view.fxml* décrivant une fenêtre

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

<?import javafx.scene.control.Label?>
<?import javafx.scene.layout.VBox?>

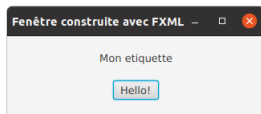
<?import javafx.scene.control.Button?>
<VBox alignment="CENTER" spacing="20.0"
  xmlns:fx="http://javafx.com/fxml"
  fx:controller="com.exemples.cours.SalutLeMondeAvecFXMLController">

  <Label text="Mon etiquette" fx:id="texteDeSalut"/>
  <Button text="Hello!" onAction="#onClicBoutonBonjour"/>
</VBox>
```

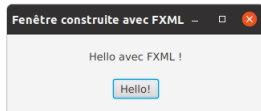
```
/* contrôleur - classe Java responsable de la
   gestion des interactions utilisateur */

public class SalutLeMondeAvecFXMLController {
    @FXML
    private Label texteDeSalut;

    @FXML
    protected void onClicBoutonBonjour() {
        texteDeSalut.setText("Hello avec FXML !");
    }
}
```



clic ↓



FXML et Contrôleur - comment ça marche ?

Dans le fichier *.fxml* :

- l'attribut "fx:id" sert à identifier les objets créés à partir du *.fxml* dans le code JavaFX
- dans le contrôleur ces objets sont référencés en tant qu'attributs de la classe contrôleur
- le référencement des méthodes du contrôleur dans le *.fxml* se fait avec le symbole #
- l'attribut fx:controller sert à désigner la classe Java qui servira de contrôleur (**un seul contrôleur par fichier .fxml**)

Dans le *contrôleur* :

- les éléments du *.fxml* sont désigné par l'annotation @FXML

FXML et Contrôleur - comment ça marche ?

L'instanciation du contrôleur a lieu automatiquement durant l'appel de la méthode `fxmlLoader.load()` ... si celui-ci possède un constructeur sans paramètres (par ex. si constructeur par défaut).

Si pas de constructeur sans paramètres dans le contrôleur :

```
public class MonContrôleur {
    // des attributs ici

    public MonContrôleur(String s1, String s2){
        // du code ici
    }

    // des méthodes ici
}
```

```
FXMLLoader fxmlLoader = new FXMLLoader(
    SalutLeMondeAvecFXML.class.getResource("hello-view.fxml"));
fxmlLoader.setControllerFactory(c -> new MonContrôleur("titi", "tata"));
Scene scene = new Scene(fxmlLoader.load(), 320, 240);
```

FXML et Contrôleur - comment ça marche ?

- Le loader FXML va automatiquement chercher les variables d'instances du contrôleur.
 1. Si elle sont publiques, alors ok (attributs publiques dans une appli O. O. ??????)
 2. Si non, alors seules celles annotées avec @FXML sont accessibles par le loader.
- Si le nom d'une variable est identique à un élément `fx:id` dans le `.fxml`, la référence de l'objet du `.fxml` est automatiquement copiée dans le contrôleur.

Une bonne pratique est d'annoter avec @FXML tous les éléments (attributs et méthodes) utilisées dans le fichier `.fxml`.

Interprétation du FXML - exemple complet



Résumé du code FXML (avec structure générale de la fenêtre) :

```
public class FullViewControler {
    @FXML
    private Label commandeValidee;

    public void clicValider() {
        // du code manipulant le label commandeValidee
        commandeValidee.setText("Commande acceptée");
        commandeValidee.setFont(new Font("Arial", 20));
    }

    public void clicFermer() {
        Platform.exit();
    }
}
```

```
<BorderPane prefHeight="400.0" prefWidth="600.0"
    style="-fx-background-color: LIGHTBLUE;"
    xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1"
    fx:controller="com.exemples.cours.fxml.FullViewControler">
    <top>
        <MenuBar BorderPane.alignment="CENTER">
            <Menu text="Accueil" />
            <Menu text="Produits" />
            <Menu text="Livraison" />
            <Menu text="FAQ" />
        </MenuBar>
    </top>
    <center>
    </center>
    <bottom>
        <VBox alignment="CENTER" prefHeight="139.0" prefWidth="600.0"
            BorderPane.alignment="CENTER">
            <Label fx:id="commandeValidee"/>
            <HBox alignment="CENTER" spacing="40.0"
                prefHeight="100.0" prefWidth="200.0"
                style="-fx-font-size: 15px">
                <Button onAction="#clicValider" text="Valider"/>
                <Button onAction="#clicFermer" text="Fermer"/>
            </HBox>
        </VBox>
    </bottom>
</BorderPane>
```

Interprétation du FXML - exemple complet

Résumé du code FXML (centre du BorderPane) :

```
<center>
  <VBox alignment="CENTER" prefHeight="200.0" prefWidth="100.0" BorderPane.alignment="CENTER">
    <Label graphicTextGap="7.0" lineSpacing="4.0" text="Mon panier :" textOverrun="WORD_ELLIPSIS">
      <font>
        <Font name="Linux Libertine Mono 0" size="17.0" />
      </font>
    </Label>
    <GridPane alignment="CENTER" minHeight="10" minWidth="100">
      <columnConstraints...>
      <CheckBox text="Pommes" GridPane.rowIndex="1" />
      <CheckBox text="Choux" GridPane.rowIndex="2" />
      <CheckBox text="Carottes" GridPane.rowIndex="3" />
      <HBox alignment="CENTER_RIGHT" prefHeight="100.0" prefWidth="200.0" spacing="20.0" GridPane.columnIndex="1" GridPane.rowIndex="1">
        <Label text="quantité en kg" />
        <Spinner editable="true" promptText="1" />
      </HBox>
      <HBox alignment="CENTER_RIGHT" prefHeight="100.0" prefWidth="200.0" spacing="20.0" GridPane.columnIndex="1" GridPane.rowIndex="2">
        <Label text="quantité" />
        <Spinner editable="true" promptText="1" />
      </HBox>
      <HBox alignment="CENTER_RIGHT" prefHeight="100.0" prefWidth="200.0" spacing="20.0" GridPane.columnIndex="1" GridPane.rowIndex="3">
        <Label text="quantité en kg" />
        <Spinner editable="true" promptText="1" />
      </HBox>
    </GridPane>
  </VBox>
</center>
```

Initialisation du contrôleur

- Si l'on souhaite initialiser certains attributs du contrôleur, il faut ajouter une méthode sans arguments nommée `initialize()`.
- Le loader FXML va automatiquement appeler cette méthode après le chargement du `.fxml` (et la construction de l'objet contrôleur).

```
public class FullViewControlerAvecInitialisation {  
    @FXML  
    private CheckBox carottes;  
    @FXML  
    private CheckBox pommes;  
    @FXML  
    private CheckBox choux;  
  
    public void initialize(){  
        carottes.setSelected(true);  
        pommes.setSelected(false);  
        choux.setSelected(true);  
    }  
}
```

