



Cours IHM-1

JavaFX

7 - FXML

SceneBuilder

I/F procédurales ↔ déclaratives



- Avec *JavaFX*, les interfaces peuvent être créées de deux manières :
 - **Procédurale** : en écrivant du code *Java* qui fait appel aux API de la plateforme et qui utilise les composants/conteneurs à disposition (classes et interfaces)
 - **Déclarative** : en décrivant l'interface dans un fichier au format FXML qui sera ensuite chargé dynamiquement dans l'application
- Les premiers chapitres du cours ont décrit les bases de la technique procédurale (programmatisation) permettant de créer des interfaces.
- Le présent chapitre abordera la deuxième possibilité de créer les interfaces (les vues) en utilisant notamment l'outil *SceneBuilder* qui permet, de manière interactive, de créer les fichiers FXML.
- *SceneBuilder* est une application qui doit être installée et, pour une utilisation avec *Eclipse*, le plugin *e(fx)clipse* est recommandé.



- Au centre de l'approche déclarative, se trouve les **fichiers FXML**.
- Un fichier FXML est un fichier au format XML dont la syntaxe est conçue pour décrire l'interface (la vue) avec ses composants, ses conteneurs, sa disposition, ...
 - Le fichier FXML 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, ...).
 - Un fichier FXML constitue une forme particulière de sérialisation d'objets, utilisée spécifiquement pour décrire les interfaces
- 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.



- Les objets créés par le chargement de fichiers FXML peuvent être assignés à la **racine d'un graphe de scène** ou représenter **un des nœuds** dans un graphe de scène créé de manière procédurale.
- Une fois chargés, les nœuds issus de fichiers FXML sont totalement équivalents à ceux créés de manière procédurale. Les mêmes opérations et manipulations peuvent leur être appliquées.
- 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**
- Même si certaines possibilités existent (en lien notamment avec du code *JavaScript*) on conseille généralement d'utiliser les fichiers FXML exclusivement pour décrire les interfaces, et d'effectuer tous les traitements (activité des contrôleurs) dans le code *Java*.

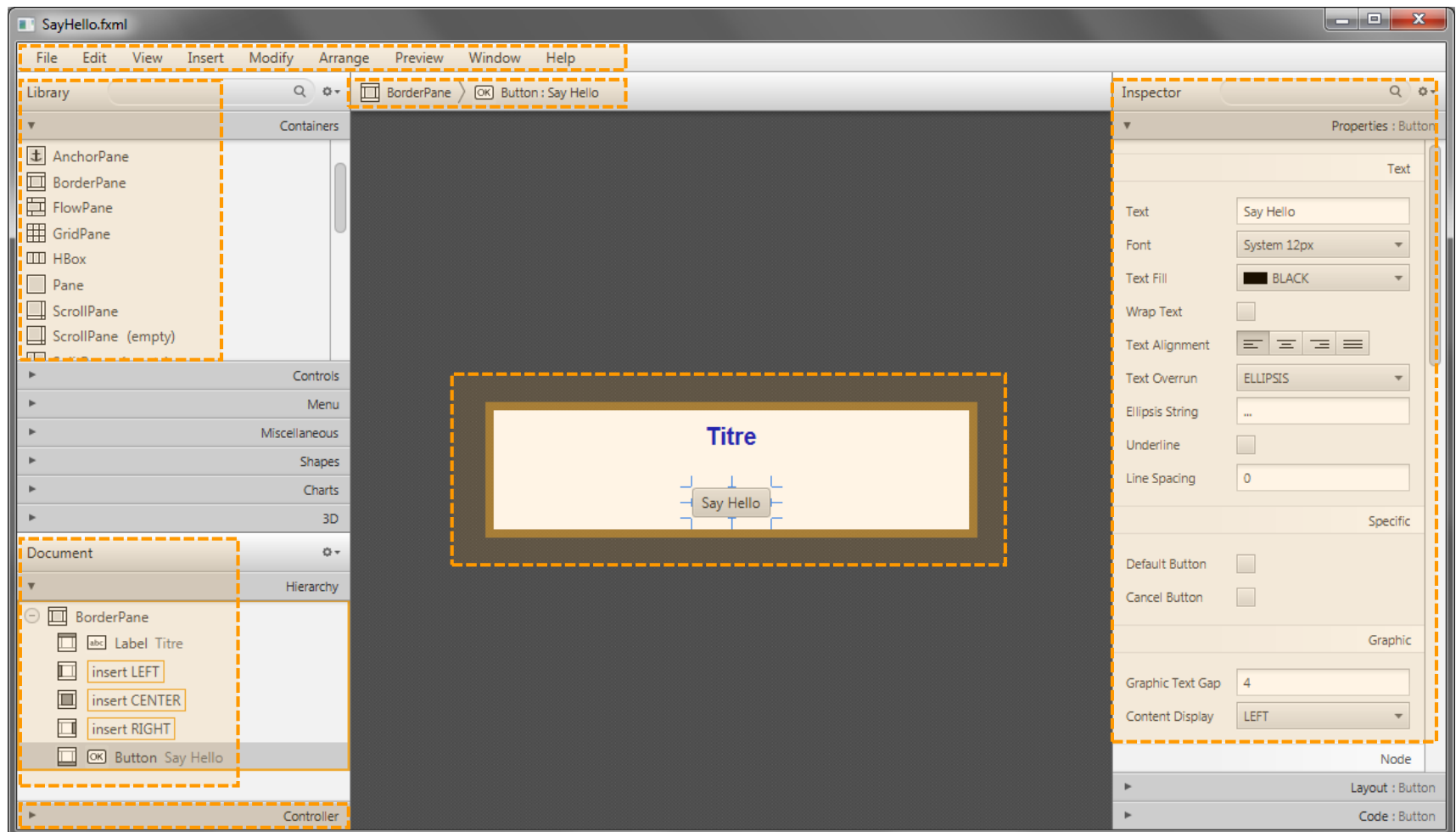


- L'outil graphique **SceneBuilder** permet de concevoir l'interface de manière interactive (*WYSIWYG*) en assemblant les conteneurs et les composants et en définissant leurs propriétés.
- Le mode de fonctionnement de cet utilitaire est assez classique avec une zone d'édition centrale, entourée d'un 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 décrit en détail dans ce support de cours, 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).
 - Malgré l'outil graphique, on n'échappe donc pas à une compréhension minimale des API (composants, conteneurs, propriétés, ...).

SceneBuilder [2]

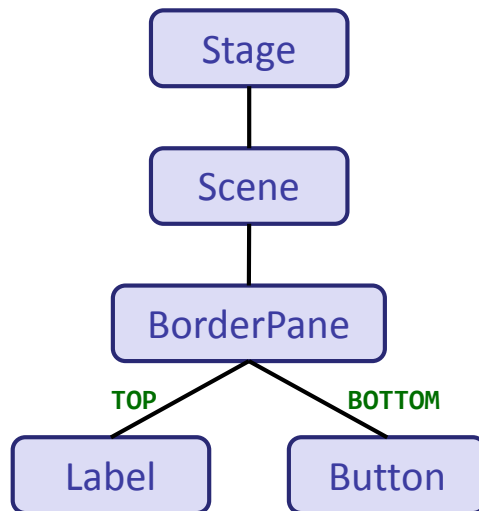


- Aperçu de l'écran principal :





- Un exemple d'application très simple :
 - Un conteneur `BorderPane`
 - Deux composants : `Label` et `Button`
 - Quelques adaptations de propriétés (taille, couleur, marge, ...)
- Graphe de scène et apparence finale de l'application





- Variante procédurale avec les API *JavaFX* :

```
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);
});
```





- La même application a été créée de manière déclarative et la vue est donc décrite dans un fichier FXML qui a été créé avec *SceneBuilder*.
- 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.



- Partie principale du fichier FXML créé avec *SceneBuilder*

```
<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.
 - Référence relative au package courant par défaut ("`views/Login.fxml`")
 - Référence absolue si `'/'` initial dans le nom ("`/resources/log/Error.fxml`") (attention: le caractère `'/'` est aussi utilisé même s'il s'agit de packages et sous-packages)

```
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();  
}
```



- Dans la variante déclarative, une classe séparée joue le rôle de contrôleur pour traiter l'action du clic sur le bouton.
- Le principe de fonctionnement et le rôle des annotations seront expliqués dans les pages qui suivent.

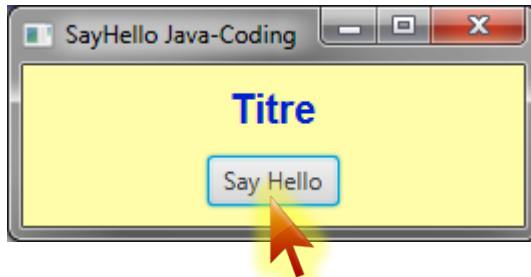
```
public class SayHelloController {  
  
    @FXML  
    private Button btnHello; // Object injected by FXMLLoader (fx:id="btnHello")  
  
    @FXML  
    private Label title;      // Object injected by FXMLLoader (fx:id="title")  
  
    @FXML  
    private void handleButtonAction(ActionEvent event) {  
        title.setText("H e l l o  !");  
        title.setTextFill(Color.FUCHSIA);  
    }  
}
```





- Les deux variantes de l'application (avec interface créée de manière procédurale et déclarative) fonctionnent de manière strictement identique.
 - Un clic sur le bouton modifie le texte et la couleur du composant `Label`.

Variante procédurale



Variante déclarative



Interprétation des fichiers FXML [1]



- 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

```
<TextField GridPane.columnIndex="3" . . .
```

l'attribut sera interprété comme une invocation de méthode statique

```
TextField tfd = new TextField();  
GridPane.setColumnIndex(tfd, 3);  
. . .
```

Interprétation des fichiers FXML [2]



- 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>
    <Font name="SansSerif Bold" size="20.0" />
  </font>
</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>`).

Liens FXML ↔ programme [1]



- Le lien entre les composants décrits dans le fichier FXML et le programme est établi par les **attributs `fx:id`** :

```
<Label id="title" fx:id="title" text="Titre" textFill="#0022cc" ...
```

- L'attribut `fx:id` fonctionne en lien avec l'annotation **`@FXML`** que l'on peut utiliser dans les contrôleurs, et qui va indiquer au système que le composant avec le nom `fx:id` pourra être *injecté* dans l'objet correspondant de la classe contrôleur.

```
public class SayHelloController {  
    @FXML  
    private Button btnHello;  
  
    @FXML                                // fx:id="title"  
    private Label title;                // Object injected by FXMLLoader  
}
```


Liens FXML ↔ programme [2]



- La classe qui joue le rôle de contrôleur pour une interface déclarée en FXML doit être annoncée **dans l'élément racine**, en utilisant l'attribut **fx:controller** :

```
<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">
```

. . .

Attention : Les attributs qui définissent les *namespaces* `xmlns=...` ainsi que `xmlns:fx=...` sont utilisés par l'environnement *JavaFX* (*FXMLLoader*, *SceneBuilder*, etc.).



Ils ne doivent donc pas être modifiés !

Liens FXML ↔ programme [3]



- Pour les **composants actifs** déclarés dans une interface en FXML, on peut indiquer la méthode du contrôleur qui doit être invoquée en utilisant l'attribut **fx:onEvent="#methodName"** :

```
<Button fx:id="btnHello" onAction="#handleButtonAction"  
        text="Say Hello" BorderPane.alignment="CENTER" />
```

. . .

- Dans la classe contrôleur, ces méthodes devront (comme les composants associés) être annotées avec **@FXML**.

```
. . .  
@FXML  
private void handleButtonAction(ActionEvent event) {  
    title.setText("H e l l o   !");  
    title.setTextFill(Color.FUCHSIA);  
}
```

Liens FXML ↔ programme [4]



- Dans les classes qui agissent comme "*contrôleurs*", on peut définir une méthode `initialize()` (qui doit être annotée avec `@FXML`) pour effectuer certaines initialisations.
- Cette méthode est automatiquement invoquée après le chargement du fichier FXML.
- Elle peut être utile pour initialiser certains composants, en faisant par exemple appel au modèle.

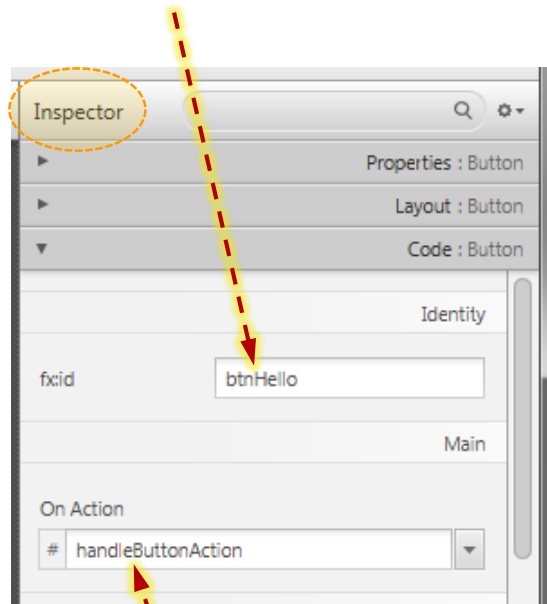
```
. . .  
@FXML  
private void initialize() {  
    cbbCountry.getItems().addAll("Allemagne", "Angleterre", "Belgique",  
                                "Espagne", "France", "Italie",  
                                "Pays-Bas", "Portugal", "Suisse");  
  
    lstProducts.getItems().addAll(model.getProducts());  
    . . .  
}
```

Liens FXML ↔ programme [5]

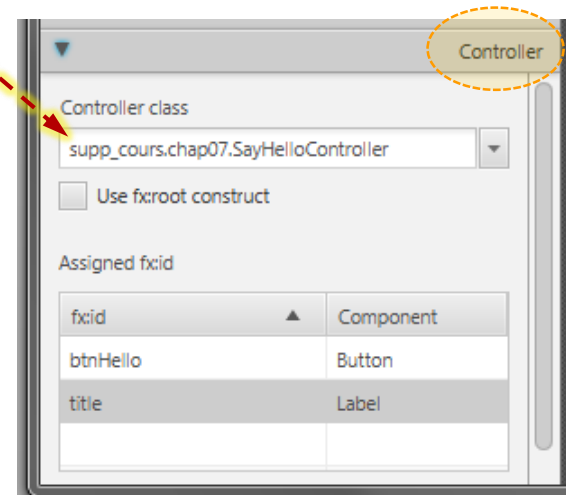


- Dans l'éditeur *SceneBuilder*, ces différents attributs de liaison avec le programme doivent être indiqués dans les champs prévus.

- Attribut `fx:id` et attribut `fx:controller`



- Attribut `fx:on...`



Le bloc *Controller* est situé en bas, à gauche de l'écran principal.

Liens FXML ↔ programme [6]



- L'attribut **id** ne doit pas être confondu avec l'attribut `fx:id`.

```
<Label id="title" fx:id="title" text="Titre" textFill="#0022cc" ...
```



- L'attribut **id** définit un *sélecteur CSS* de type **Id** qui permet d'associer un style aux composants portant cet *Id*.

- Exemple dans un fichier CSS :

```
#title {  
    -fx-font-size: 24pt;  
}
```

- Dans le fichier FXML, une **feuille de style** (fichier CSS) peut être associé à un composant avec l'attribut `stylesheets="@CSS_File"`

```
<BorderPane stylesheets="@SayHello.css" . . . >
```

- L'éditeur *SceneBuilder* permet (dans l'inspecteur de propriétés) de créer l'attribut **id** et de définir le fichier CSS associé (*Stylesheets*).

Liens FXML ↔ programme [7]



- Il est possible d'accéder au **contrôleur** associé au fichier FXML en créant un chargeur (*loader*) pour ce fichier (plutôt que d'utiliser la méthode statique `FXMLLoader.load()`).
- Cela peut être utile pour avoir accès au contrôleur, par exemple pour lui communiquer la référence du modèle de l'application :

```
public void start(Stage primaryStage) throws Exception {  
    //--- Chargement du fichier FXML et recherche du contrôleur associé  
    FXMLLoader loader = new FXMLLoader(getClass().getResource("SayHello.fxml"));  
  
    BorderPane      root = loader.load();  
    SayHelloController ctrl = loader.getController();  
    ctrl.setModel(model);  
    Scene scene = new Scene(root);  
    . . .  
}
```

- Dans ce cas, on utilisera la méthode d'instance `load()` pour charger le fichier FXML et obtenir la référence de la racine du graphe de scène.

Liens FXML ↔ programme [8]



- Si le **contrôleur** d'une interface déclarée avec FXML ne possède **pas de constructeur par défaut**, il faut créer le contrôleur et l'associer dans le code du programme, avant le chargement du fichier FXML.
- Le code suivant illustre la manière de le faire :

```
public void start(Stage primaryStage) throws Exception {  
  
    //--- Chargement du fichier FXML et association du contrôleur  
    FXMLLoader loader = new FXMLLoader(getClass().getResource("SayHello.fxml"));  
    loader.setController(new SayHelloController("a param"));  
    BorderPane root = loader.load();  
  
    . . .  
}
```

- Dans ce cas, il n'est pas nécessaire de déclarer le contrôleur dans le fichier FXML (`fx:controller="..."`).