

BUT Informatique 1^{ère} année

R2.02: Développement d'applications avec IHM

Cours 5 : JavaFX

Sébastien Lefèvre
sebastien.lefeuvre@univ-ubs.fr

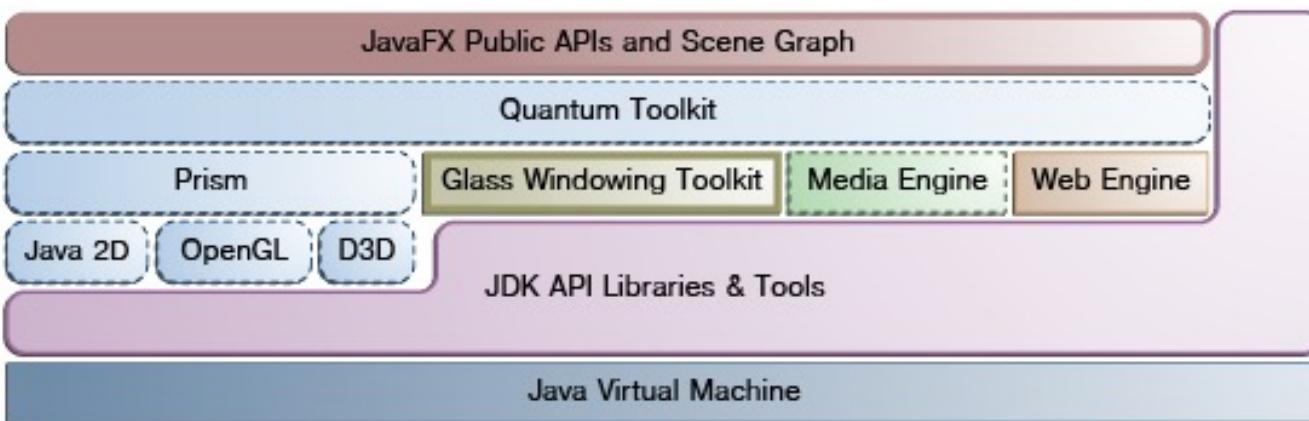
JavaFX

Introduit en 2008 pour remplacer AWT et Swing
JavaFX 2.0 depuis 2011

Open source depuis JDK 11 (à télécharger depuis openjfx.io)
Sinon, intégré à Java SE 8 (inclus dans JDK/JRE)

Composé de plusieurs éléments :

- Prism: moteur graphique
- Glass: système de fenêtrage
- Moteur web



HelloWorld

```
package helloworld;

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 HelloWorld extends Application {
    @Override
    public void start(Stage primaryStage) {
        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);
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

A retenir

- La classe principale d'une application JavaFX hérite de `javafx.application.Application`
- La méthode `start()` est le point d'entrée principal
- L'interface utilisateur est définie par deux niveau : *stage* et *scene*
 - La classe Stage est le conteneur de plus haut niveau principal
 - La classe Scene est le conteneur pour tout le contenu
- Le contenu d'une scène est représenté par un graphe hiérarchique de nœuds (chaque nœud est un composant graphique)
- La méthode `main()` n'est pas nécessaire lorsque le JAR est créé avec JavaFX Packager, mais reste nécessaire sinon.
Il est donc utile de conserver une méthode main() pour exécuter le JAR dans tous les cas.

1^{er} exemple : HelloWorld

```
package helloworld;

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 HelloWorld extends Application {
    @Override
    public void start(Stage primaryStage) {
        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);
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

2ème exemple : HelloFX

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloFX extends Application {

    @Override
    public void start(Stage stage) {
        String javaVersion = System.getProperty("java.version");
        String javafxVersion = System.getProperty("javafx.version");
        Label l = new Label("Hello, JavaFX " + javafxVersion + ", running on Java " + javaVersion + ".");
        Scene scene = new Scene(new StackPane(l), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

Pas trop de changement...

- Gestionnaires de placement
- Composants : Text, Label, TextField, PasswordField, Button, ...
- Réaction aux évènements :
`addActionListener ➔ setOnAction`

...

Mais si quand même !

- Utilisation d'un CSS dans le code JavaFX

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

- Identification de composants

```
scenetitle.setId("welcome-text");  
actiontarget.setId("actiontarget");
```

Tout le reste se fait dans le CSS...

Login.css

```
.root { -fx-background-image: url("background.jpg"); }

.label { -fx-font-size: 12px; -fx-font-weight: bold; -fx-text-fill: #333333; -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 ); }

#welcome-text { -fx-font-size: 32px; -fx-font-family: "Arial Black"; -fx-fill: #818181; -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6, 0.0 , 0 , 2 ); }

#actiontarget { -fx-fill: FIREBRICK; -fx-font-weight: bold; -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 ); }

.button { -fx-text-fill: white; -fx-font-family: "Arial Narrow"; -fx-font-weight: bold; -fx-background-color: linear-gradient(#61a2b1, #2A5058); -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 ); }

.button:hover { -fx-background-color: linear-gradient(#2A5058, #61a2b1); }
```

Login.java

```
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import static javafx.geometry.HPos.RIGHT;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class Login extends Application {
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("JavaFX Welcome");
        GridPane grid = new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(25, 25, 25, 25));
        Text scenetitle = new Text("Welcome");
        scenetitle.setId("welcome-text");
        grid.add(scenetitle, 0, 0, 2, 1);
        Label userName = new Label("User Name:");
        grid.add(userName, 0, 1);
        TextField userTextField = new TextField();
        grid.add(userTextField, 1, 1);
        Label pw = new Label("Password:");
        grid.add(pw, 0, 2);
        PasswordField pwBox = new PasswordField();
        grid.add(pwBox, 1, 2);
        Button btn = new Button("Sign in");
        HBox hbBtn = new HBox(10);
        hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
        hbBtn.getChildren().add(btn);
        grid.add(hbBtn, 1, 4);
        final Text actiontarget = new Text();
        grid.add(actiontarget, 0, 6);
        grid.setColumnSpan(actiontarget, 2);
        grid.setAlignment(actiontarget, RIGHT);
        actiontarget.setId("actiontarget");
        btn.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent e) {
                actiontarget.setText("Sign in button pressed");
            }
        });
        Scene scene = new Scene(grid, 300, 275);
        scene.getStylesheets()
            .add(Login.class.getResource("Login.css")
            .toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

Isoler davantage la vue...

- La définition du contenu de l'IHM peut être faite en mode déclaratif dans un fichier XML

(pour plus de détails : https://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html)

```
BorderPane border = new BorderPane();
Label toppanetext = new Label("Page Title");
border.setTop(toppanetext);
Label centerpanetext = new Label ("Some data here");
border.setCenter(centerpanetext);
```



```
<BorderPane>
  <top>
    <Label text="Page Title"/>
  </top>
  <center>
    <Label text="Some data here"/>
  </center>
</BorderPane>
```

- Le rendu visuel est fait dans le CSS
- On conserve en Java uniquement le code de réaction aux évènements et la logique applicative

FXML Example.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import java.net.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<GridPane fx:controller="fxmlexample.FXMLEExampleController"
xmlns:fx="http://javafx.com/fxml" alignment="center"
hgap="10" vgap="10" styleClass="root">
<padding>
    <Insets top="25" right="25" bottom="10" left="25"/>
</padding>
<Text id="welcome-text" text="Welcome"
    GridPane.columnIndex="0" GridPane.rowIndex="0"
    GridPane.columnSpan="2"/>
<Label text="User Name:"
    GridPane.columnIndex="0" GridPane.rowIndex="1" />
<TextField
    GridPane.columnIndex="1" GridPane.rowIndex="1" />
<Label text="Password:"
    GridPane.columnIndex="0" GridPane.rowIndex="2" />
<PasswordField fx:id="passwordField"
    GridPane.columnIndex="1" GridPane.rowIndex="2" />
<HBox spacing="10" alignment="bottom_right"
    GridPane.columnIndex="1" GridPane.rowIndex="4">
    <Button text="Sign In"
        onAction="#handleSubmitButtonAction" />
</HBox>
<Text fx:id="actiontarget" GridPane.columnIndex="0"
    GridPane.columnSpan="2" GridPane.halignment="RIGHT"
    GridPane.rowIndex="6" />
<stylesheets>
    <URL value="@Login.css" />
</stylesheets>
</GridPane>
```

Login.CSS

```
.root { -fx-background-image: url("background.jpg"); }

.label { -fx-font-size: 12px; -fx-font-weight: bold; -fx-text-fill: #333333; -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 ); }

#welcome-text { -fx-font-size: 32px; -fx-font-family: "Arial Black"; -fx-fill: #818181; -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6, 0.0 , 0 , 2 ); }

#actiontarget { -fx-fill: FIREBRICK; -fx-font-weight: bold; -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 ); }

.button { -fx-text-fill: white; -fx-font-family: "Arial Narrow"; -fx-font-weight: bold; -fx-background-color: linear-gradient(#61a2b1, #2A5058); -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 ); }

.button:hover { -fx-background-color: linear-gradient(#2A5058, #61a2b1); }
```

FXMLExample.java

```
package fxmlexample;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class FXMLExample
    extends Application {
    @Override
    public void start(Stage stage)
        throws Exception {
        Parent root =
            FXMLLoader.load(getClass()
                .getResource("fxml_example.fxml"));
        stage.setTitle("FXML Welcome");
        stage.setScene(new Scene(root, 300, 275));
        stage.show();
    }

    public static void main(String[] args) {
        Application.launch(FXMLExample.class, args);
    }
}
```

FXMLExampleController.java

```
package fxmlexample;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.text.Text;
public class FXMLExampleController {

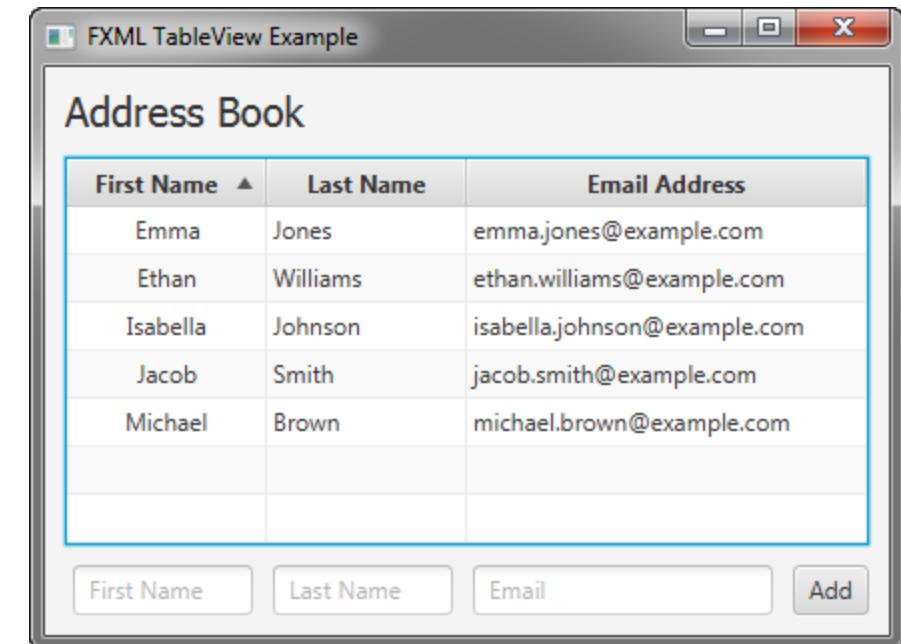
    @FXML
    private Text actiontarget;

    @FXML
    protected void handleSubmitButtonAction(ActionEvent event) {
        actiontarget.setText("Sign in button pressed");
    }
}
```

The `@FXML` annotation is used to **tag nonpublic controller member fields and handler methods for use by FXML markup.**

Autre exemple

<https://docs.oracle.com/javase/8/javafx/sample-apps/FXMLTableView.zip>



FXMLTableView.java

```
package fxmltableview;  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.scene.layout.Pane;  
import javafx.stage.Stage;  
public class FXMLTableView extends Application {  
    @Override public void start(Stage primaryStage) throws Exception {  
        primaryStage.setTitle("FXML TableView Example");  
        Pane myPane = (Pane) FXMLLoader.  
            load(getClass().getResource("fxml_tableview.fxml"));  
        Scene myScene = new Scene(myPane);  
        primaryStage.setScene(myScene);  
        primaryStage.show();  
    }  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

FXMLTableviewController.java

```
package fxmltableview;

import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;

public class FXMLTableViewController {
    @FXML private TableView<Person> tableView;
    @FXML private TextField firstNameField;
    @FXML private TextField lastNameField;
    @FXML private TextField emailField;

    @FXML
    protected void addPerson(ActionEvent event) {
        ObservableList<Person> data = tableView.getItems();
        data.add(new Person(firstNameField.getText(),
                            lastNameField.getText(),
                            emailField.getText())
        );
        firstNameField.setText("");
        lastNameField.setText("");
        emailField.setText("");
    }
}
```

Person.java

```
package fxmltableview;
import javafx.beans.property.SimpleStringProperty;

public class Person {

    private final SimpleStringProperty firstName = new SimpleStringProperty("");
    private final SimpleStringProperty lastName = new SimpleStringProperty("");
    private final SimpleStringProperty email = new SimpleStringProperty("");

    public Person() {this("", "", "")}

    public Person(String firstName, String lastName, String email) {
        setFirstName(firstName);
        setLastName(lastName);
        setEmail(email);
    }

    public String getFirstName() {return firstName.get();}

    public void setFirstName(String fName) {firstName.set(fName);}

    public String getLastname() {return lastName.get();}

    public void setLastName(String fName) {lastName.set(fName);}

    public String getEmail() {return email.get();}

    public void setEmail(String fName) {email.set(fName);}
}
```

FormattedTableCellFactory.java

```
package fxmltableview;

import java.text.Format;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.control.TableCell;
import javafx.scene.control.TableColumn;
import javafx.scene.text.TextAlignment;
import javafx.util.Callback;

public class FormattedTableCellFactory<S, T> implements
Callback<TableColumn<S, T>, TableCell<S, T>> {

    private.TextAlignment alignment;
    private Format format;

    public.TextAlignment getAlignment() { return alignment; }

    public void setAlignment(TextAlignment alignment) {
        this.alignment = alignment; }

    public Format getFormat() { return format; }

    public void setFormat(Format format) { this.format = format; }

    @Override
    @SuppressWarnings("unchecked")
    public TableCell<S, T> call(TableColumn<S, T> p) {
        TableCell<S, T> cell = new TableCell<S, T>() {
            @Override
            public void updateItem(Object item, boolean empty) {
                if (item == getItem()) {
                    return;
                }
                super.updateItem((T) item, empty);
                if (item == null) {
                    super.setText(null);
                    super.setGraphic(null);
                } else if (format != null) {
                    super.setText(format.format(item));
                } else if (item instanceof Node) {
                    super.setText(null);
                    super.setGraphic((Node) item);
                } else {
                    super.setText(item.toString());
                    super.setGraphic(null);
                }
            }
            cell.setTextAlignment(alignment);
            switch (alignment) {
                case CENTER:
                    cell.setAlignment(Pos.CENTER);
                    break;
                case RIGHT:
                    cell.setAlignment(Pos.CENTER_RIGHT);
                    break;
                default:
                    cell.setAlignment(Pos.CENTER_LEFT);
                    break;
            }
        };
        return cell;
    }
}
```

FXML_Tableview.fxml

```
<?import javafx.collections.*?>
<?import javafx.geometry.Insets?>
<?import java.lang.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.control.cell.*?>
<?import javafx.scene.layout.*?>
<?import fxmltableview.*?>
<GridPane alignment="center" hgap="10.0" vgap="10.0"
fx:controller="fxmltableview.FXMLTableViewController"
xmlns:fx="http://javafx.com/fxml">

<padding>
  <Insets bottom="10.0" left="10.0" right ="10.0" top="10.0"/>
</padding>

<Label text="Address Book" GridPane.columnIndex="0 » GridPane.rowIndex="0"
style="--fx-font: NORMAL 20 Tahoma;"/>

<TableView fx:id="tableView" GridPane.columnIndex="0" GridPane.rowIndex="1">
<columns>
  <TableColumn fx:id="firstNameColumn" text="First Name"
    <cellValueFactory>
      <PropertyValueFactory property="firstName" />
    </cellValueFactory>
    <cellFactory>
      <FormattedTableCellFactory alignment="center"></FormattedTableCellFactory>
    </cellFactory>
  </TableColumn>
  <TableColumn text="Last Name" prefWidth="100">
    <cellValueFactory>
      <PropertyValueFactory property="lastName" />
    </cellValueFactory>
  </TableColumn>
  <TableColumn text="Email Address" prefWidth="200">
    <cellValueFactory>
```

```
      < PropertyValueFactory property="email" />
    </cellValueFactory>
  </TableColumn>
</columns>
<items>
<FXCollections fx:factory="observableArrayList">
<Person firstName="Jacob" lastName="Smith"
  email="jacob.smith@example.com"/>
<Person firstName="Isabella" lastName="Johnson"
  email="isabella.johnson@example.com"/>
<Person firstName="Ethan" lastName="Williams"
  email="ethan.williams@example.com"/>
<Person firstName="Emma" lastName="Jones"
  email="emma.jones@example.com"/>
<Person firstName="Michael" lastName="Brown"
  email="michael.brown@example.com"/>
</FXCollections>
</items>
<sortOrder>
  <fx:reference source="firstNameColumn"/>
</sortOrder>
</TableView>

<HBox spacing="10" alignment="bottom_right" GridPane.columnIndex="0"
  GridPane.rowIndex="2">
<TextField fx:id="firstNameField" promptText="First Name" prefWidth="90"/>
<TextField fx:id="lastNameField" promptText="Last Name" prefWidth="90"/>
<TextField fx:id="emailField" promptText="Email" prefWidth="150"/>
<Button text="Add" onAction="#addPerson"/>
</HBox>
</GridPane>
```



: de Swing à JavaFX...

iutvannes.fr

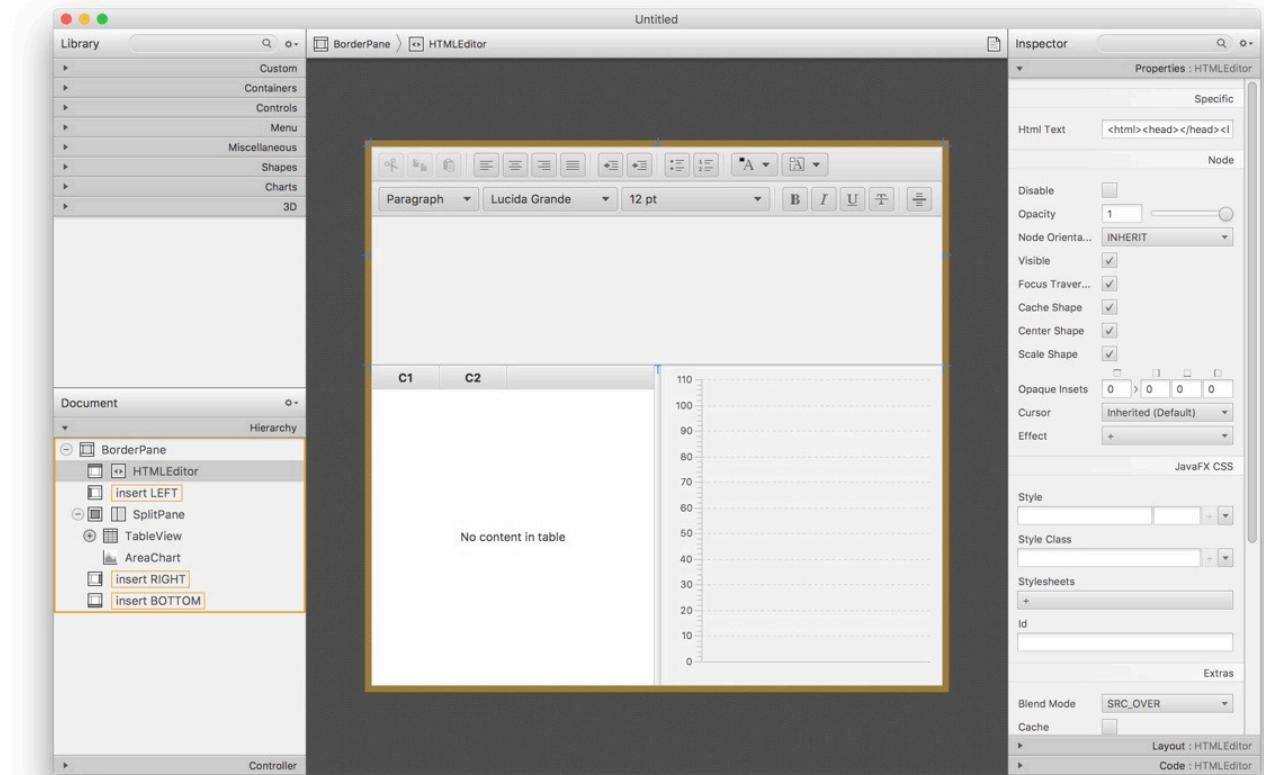
Pourquoi JavaFX ?

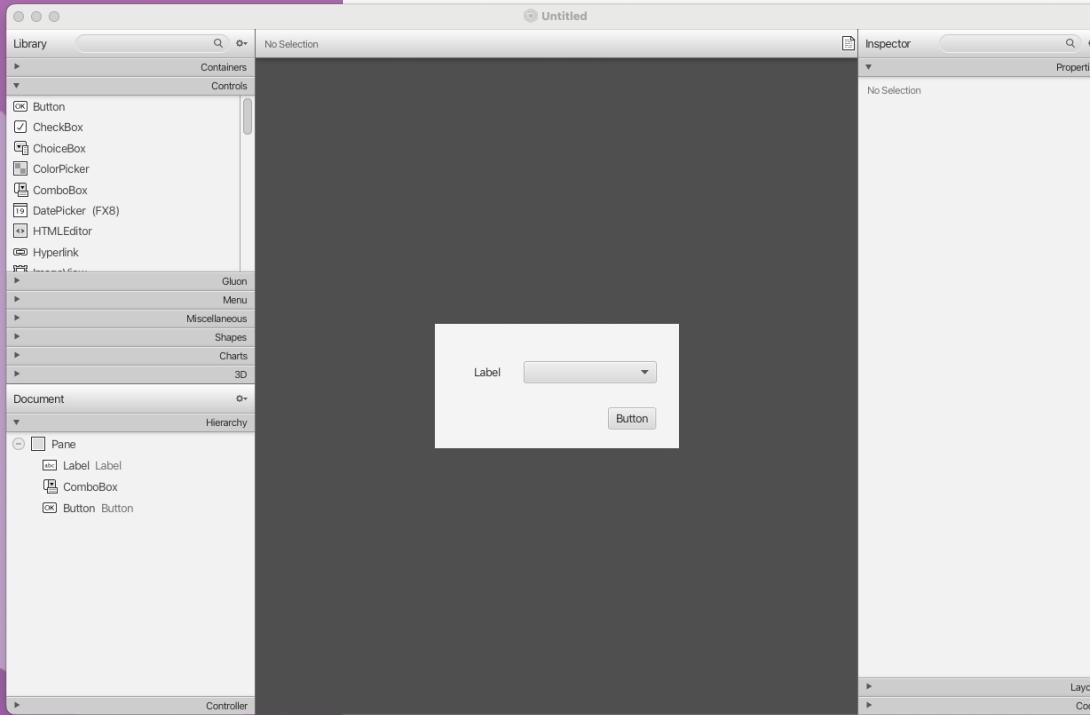
- Swing ne permet pas de développer des IHM modernes
- FXML permet de séparer la création de l'IHM d'une application JavaFX en la séparant complètement de la partie métier,
et définir une IHM avec une approche déclarative en XML est plus simple
- JavaFX Scene Builder est un outil graphique qui simplifie la conception
d'une IHM (génération automatique de code FXML)
- Support des CSS pour séparer l'apparence visuelle de la logique
applicative
- Ajout simplifié de média (audio, vidéo, ...)
- Animations de meilleure qualité avec un double buffering
- Rendu visuel de contenu HTML

et JavaFX/Gluon fonctionne aussi sur mobile (iOS/Android/Raspberry Pi)

JavaFX Scene Builder

- Outil graphique (WYSIWYG)
- Génération automatique du code FXML
- Support du CSS
- Mode prévisualisation
- Intégration dans l'IDE NetBeans





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

<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="140.0" prefWidth="275.0" xmlns="http://javafx.com/javafx/18"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Label layoutX="44.0" layoutY="46.0" text="Label" />
        <ComboBox layoutX="100.0" layoutY="42.0" prefWidth="150.0" />
        <Button layoutX="195.0" layoutY="94.0" mnemonicParsing="false" text="Button" />
    </children>
</Pane>
```

Intégrer du contenu JavaFX dans une GUI Swing

```
import javafx.application.Platform;
import javafx.embed.swing.JFXPanel;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
public class Test {
    private static void initAndShowGUI() {
        // This method is invoked on the EDT thread
        JFrame frame = new JFrame("Swing and JavaFX");
        final JFXPanel fxPanel = new JFXPanel();
        // implicitly starts the JavaFX runtime
        frame.add(fxPanel);
        frame.setSize(300, 200);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    Platform.runLater(new Runnable() {
        @Override
        public void run() {
            initFX(fxPanel);
        }
    });
}
private static void initFX(JFXPanel fxPanel) {
    // This method is invoked on the JavaFX thread
    Scene scene = createScene();
    fxPanel.setScene(scene);
}
private static Scene createScene() {
    Group root = new Group();
    Scene scene = new Scene(root, Color.ALICEBLUE);
    Text text = new Text();
    text.setX(40);
    text.setY(100);
    text.setFont(new Font(25));
    text.setText("Welcome JavaFX!");
    root.getChildren().add(text);
    return scene;
}
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            initAndShowGUI();
        }
    });
}
```

Intégrer du contenu JavaFX dans une GUI Swing

- Modifier du contenu JavaFX après un évènement Swing

```
jbutton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        Platform.runLater(new Runnable() {  
            @Override  
            public void run() {  
                fxlabel.setText("Swing button clicked!");  
            }  
        });  
    }  
});
```

JavaFX data should be accessed only on the JavaFX User thread.
Whenever you must change JavaFX data, wrap your code into a Runnable object and call the Platform.runLater method

Intégrer du contenu JavaFX dans une GUI Swing

- Modifier du contenu Swing après un évènement JavaFX

```
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                //Code to change Swing data.
            }
        });
    }
});
```

Swing data should be changed only on the EDT (Event Dispatch Thread).
To ensure that your code is implemented on the EDT, wrap it into a Runnable object and call the SwingUtilities.invokeLater method

Intégrer du contenu JavaFX dans une GUI Swing

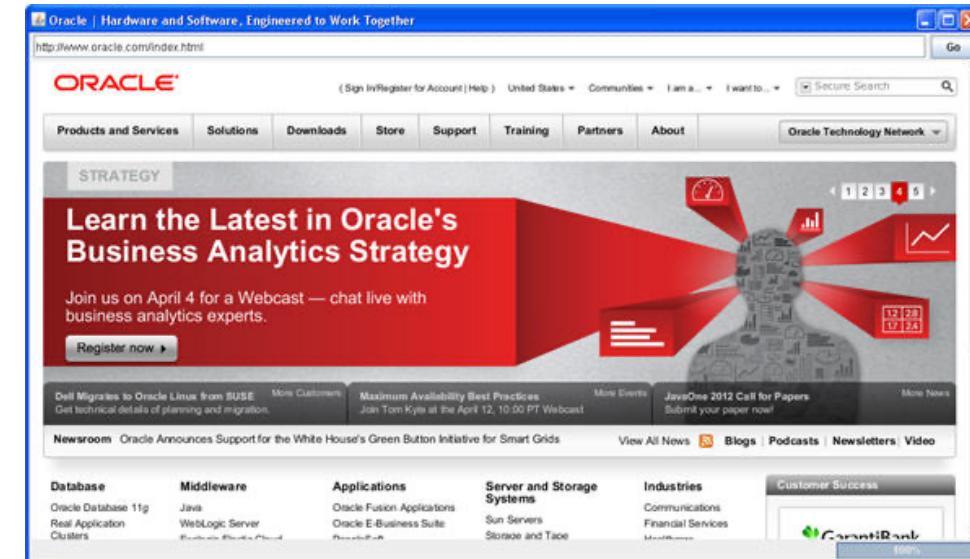
Exemple plus complet : navigateur web en Java

<https://docs.oracle.com/javase/8/javafx/interoperability-tutorial/simpleswingbrowserjava.htm#CBBHEAll>

```
export PATH_TO_FX=../javafx-sdk-18.0.1/lib
```

```
javac -d class --module-path $PATH_TO_FX  
--add-modules javafx.controls,javafx.swing,javafx.web  
src/simpleswingbrowser/SimpleSwingBrowser.java
```

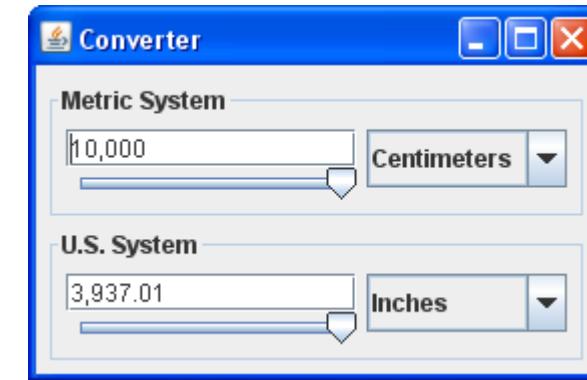
```
java -cp class --module-path $PATH_TO_FX  
--add-modules javafx.controls,javafx.swing,javafx.web  
simpleswingbrowser.SimpleSwingBrowser
```



Migrer une application Swing en JavaFX

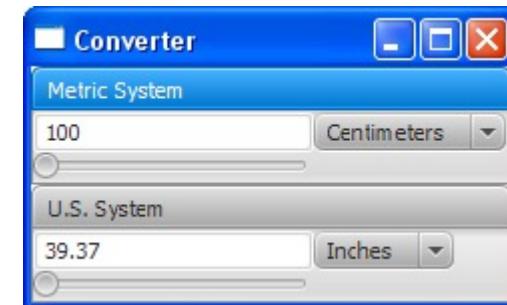
Exemple en Swing

<https://docs.oracle.com/javase/tutorial/uiswing/examples/components-ConverterProject.zip>



Exemple en JavaFX

<https://docs.oracle.com/javafx/2/swing/Converter.zip>



Application

```
public class Converter {  
    private void initAndShowGUI() {  
        ...  
    }  
    public static void main(String[] args) {  
        SwingUtilities.invokeLater(new Runnable() {  
            @Override  
            public void run() {  
                initAndShowGUI();  
            }  
        });  
    }  
}
```

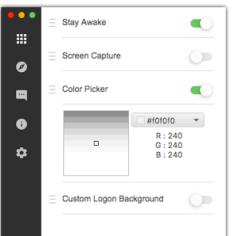
```
import javafx.application.Application;  
import javafx.stage.Stage;  
  
public class Converter extends Application {  
    @Override  
    public void start(Stage t) {  
        ...  
    }  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

Les éléments à modifier

- Gestionnaires de placements :
BorderPane, HBox/VBox, StackPane, GridPane, FlowPane,
TilePane, AnchorPane
- Composants
- Actions

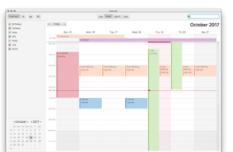
En résumé

- JavaFX plus riche que Swing: web, animation/3D, media, multi-touch, etc
- JavaFX conserve le principe de prog évènementielle et d'IHM basée sur des composants...
- mais la description de l'IHM peut être déportée dans un fichier FXML, son visuel dans un fichier CSS (et la logique reste en Java)
- Beaucoup de Component Swing ont un équivalent Control JavaFX (enlever le J, Dialog/Alert pour JOptionPane, ...
attention : List/TableView fonctionne différemment de JList/JTable)
- En JavaFX, le gestionnaire de placement est directement intégré dans le conteneur (XXXLayout → XXXPane)
- JavaFX est maintenant open source (openjfx), de nombreux composants sont disponibles...



Actlist

JavaFX Utility Platform to easy and simply execute your own act list.



CalendarFX

A Java framework for creating sophisticated calendar views



AsciidocFX

An AsciiDoc editor to build PDF, Epub, Mobi and HTML books, documents and slides



BootstrapFX

Twitter's Bootstrap CSS for JavaFX



FXyz project

3D Visualization and Components library



Gluon Maps

Tiles based geo-location map framework



Hero 1.0

CAD application



Ikonli

Font icon packs for JavaFX applications



JFX Central

Community-driven webpage about JavaFX. Runs with JavaFX and JPro on the Web.



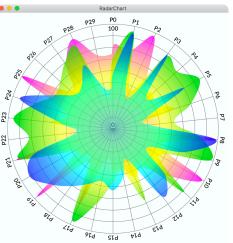
JITWatch

A log analyser and visualiser for the HotSpot JIT compiler.



Controls FX

De-facto JavaFX controls library



Charts

A library for scientific charts in JavaFX.



MaterialFX

MaterialFX is not just another theme. MaterialFX brings restyles, remade and brand new controls to JavaFX. It also offer many utilities to ease JavaFX apps development.



Medusa

A JavaFX library for Gauges and Clocks.



DSTE

The Deep Space Trajectory Explorer



FlexGanttFX

A library for rendering Gantt charts in JavaFX



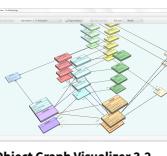
FormsFX

A framework for easily creating forms for a JavaFX application.



Modelus X

A freely available application used worldwide that enables students and teachers (high school and university) to use mathematics to create or explore models interactively.



Object Graph Visualizer 3.2

A tool for to understand the Object Oriented paradigm and patterns



Recaf

An easy to use modern Java bytecode editor



FXGL

JavaFX game engine



FXSkins

A library of new Skins for JavaFX controls. These Skins will add more functionality to the controls of your applications with no need to make code changes.



ValidatorFX

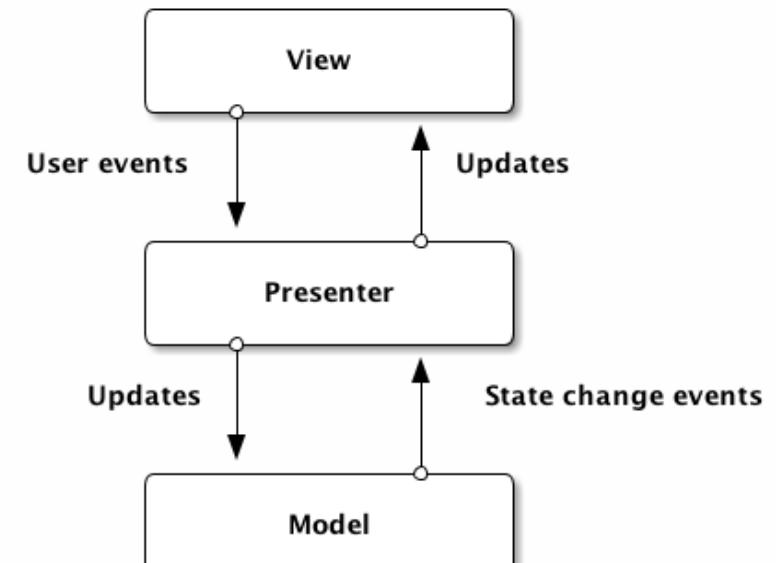
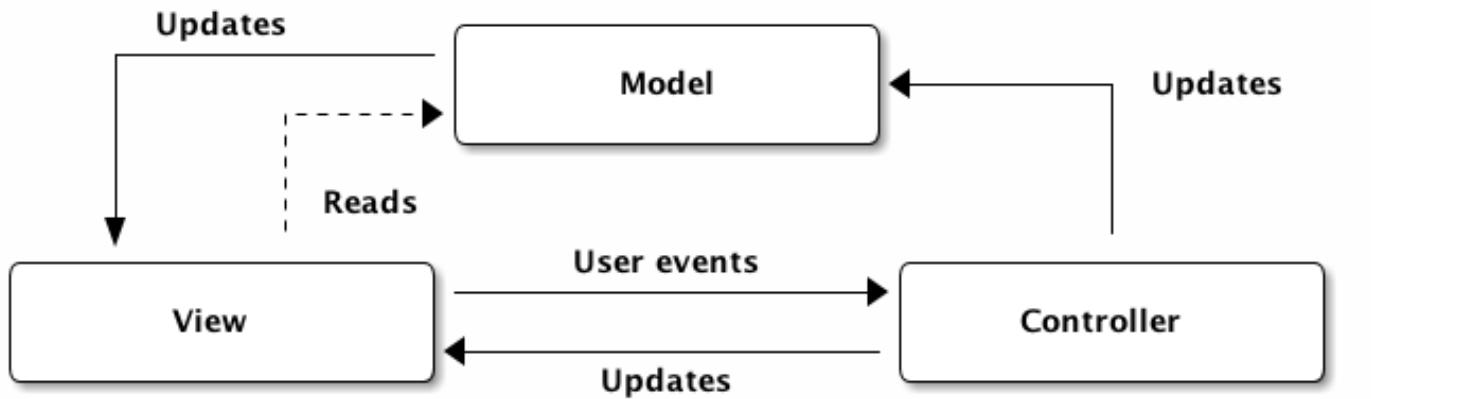
A JavaFX library containing tiles for Dashboards



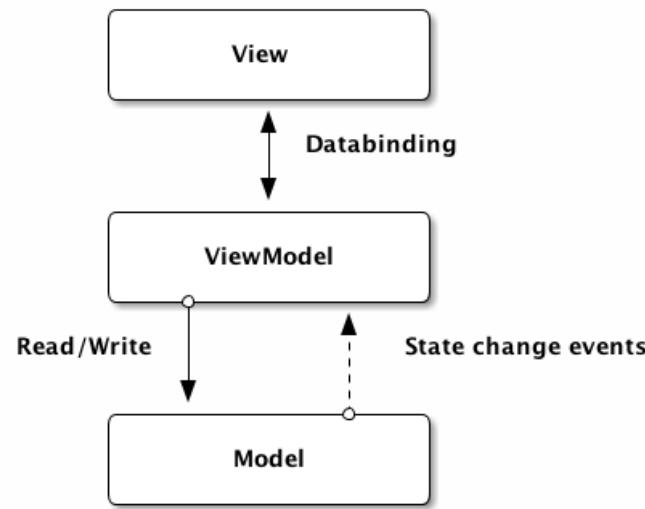
XR3Player

Powerful JavaFX Media Player + Web Browser.

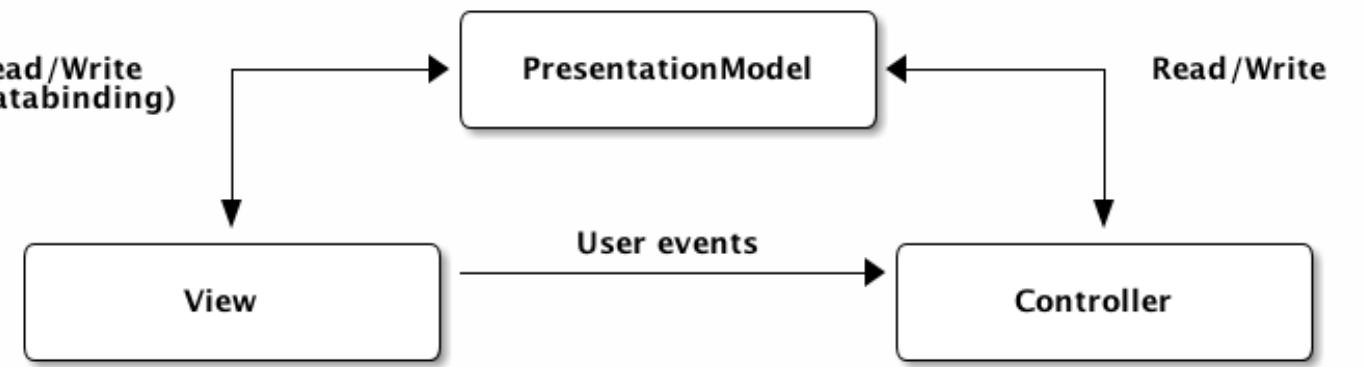
Pour finir, quelques mots sur MVC...



MVP: Model-View-Presenter



MVVM: Model-View-ViewModel



PMVC: PresentationModel-View-Controller