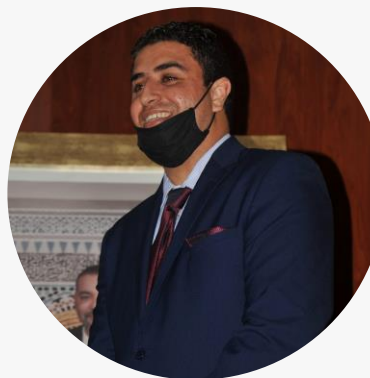


# Les interfaces graphiques avec JavaFX



**Abdelmajid BOUSSELHAM**

Email: [bousselham@enset-media.ac.ma](mailto:bousselham@enset-media.ac.ma)

Researchgate : [https://www.researchgate.net/profile/Abdelmajid\\_Bousselham2](https://www.researchgate.net/profile/Abdelmajid_Bousselham2)

Google Scholar: <https://scholar.google.com/citations?user=EZ7oxLMAAAAJ&hl=fr>

Scopus: <https://www.scopus.com/authid/detail.uri?authorId=8657730200>

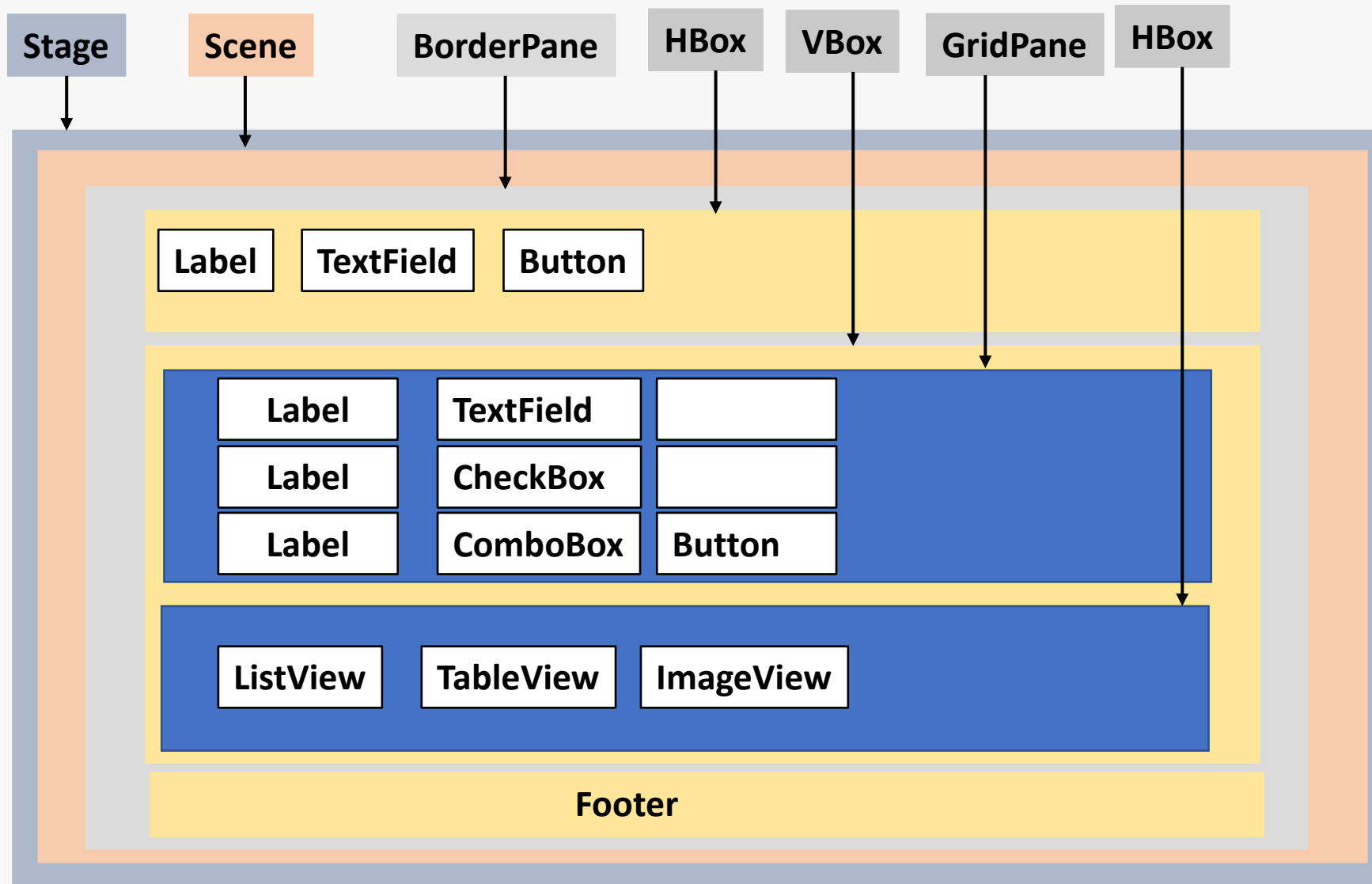
Linkedin: <https://www.linkedin.com/in/abdelmajid-bousselham-ph-d-6729341b8/>

- JavaFx est la dernière version de bibliothèques qui permet de créer des interfaces graphiques de qualité pour les applications Java Desktop, Mobile et Web.
- Avec l'apparition de Java 8 en mars 2014, JavaFX devient la bibliothèque de création de l'interface graphique officielle du langage Java .
- Le développement de son précédent SWING étant abandonné (Sauf pour les correction du bogues ) .

# Structure d'une application JavaFX

- Une application JavaFX se compose d'une hiérarchie de composants :
  - Un composant Stage qui représente la fenêtre principale de l'application. à un instant donné, le composant Stage affiche une scène.
  - Le composant Scene qui permet d'afficher tout ce qui devrait apparaître dans l'application. L'objet Scene contient des composants graphiques organisés d'une manière hiérarchique.
  - Les composants graphiques sont des objets qui peuvent être de différents types :
    - Des éléments de contrôles utilisateur : Label, TextField, ListView.
    - Des formes graphiques : Circle, Rectangle, Line ...
    - Des médias : ImageView, MediaView, etc...
    - Les Layouts pour grouper les éléments pour assurer les mises en page: BorderPane, Hbox, VBox, GridPane,...

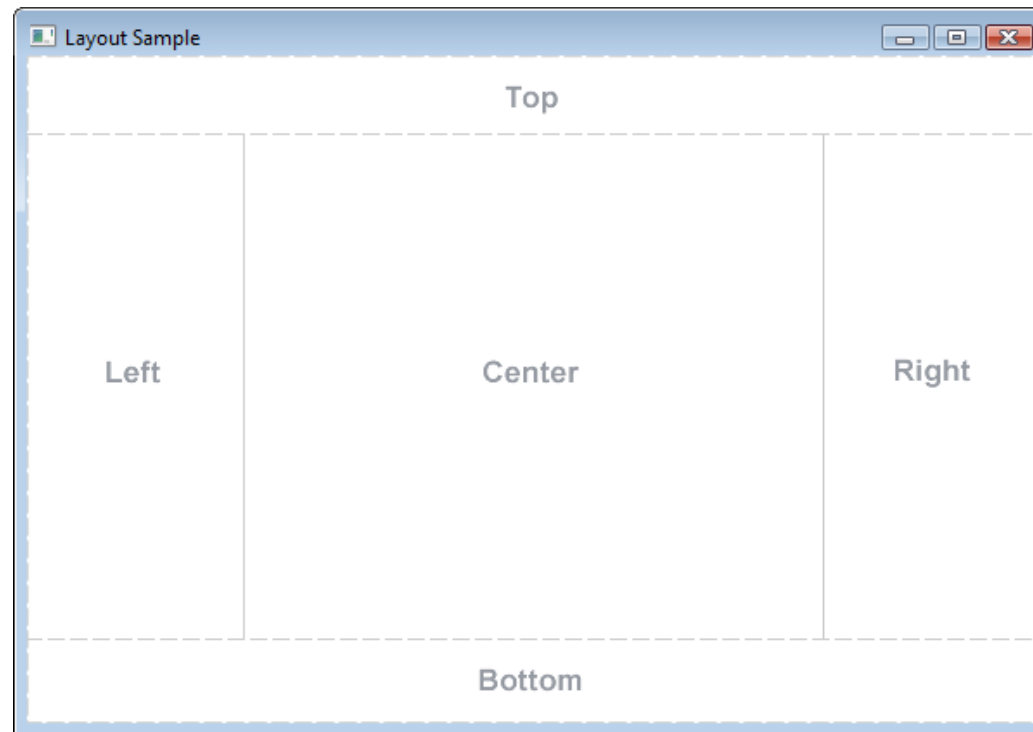
# Structure d'une application JavaFX



- Un layout ou gestionnaire de mise en page est un nœud graphique qui hérite de la classe `javafx.scene.layout.Pane`.
- Il s'agit d'une entité qui contient d'autres nœuds et qui est chargée de les déplacer, de les disposer, voire de les redimensionner de manière à changer la présentation de cet ensemble de nœuds et à les rendre utilisables dans une interface graphique.
- Un layout (organisation) est une classe de l'API Graphique permettant d'organiser les objets graphiques dans la fenêtre.

# Layout – Gestionnaire de la mise en page

- En JavaFX, il existe huit types de layouts :
  - **BorderPane** : est un conteneur (container), qui est divisé en 5 zones distinctes (Top, Bottom, Left, Right), chacune des zones peut contenir un sous-composant.



## Layout – Gestionnaire de la mise en page

- **Hbox** : est un conteneur, qui arrange les éléments graphiques d'une manière horizontale.
- **VBox** : est un conteneur, qui arrange les éléments graphiques d'une manière verticale.
- **StackPane** : est un conteneur , qui permet de ranger les éléments de façon à ce que chaque nouvel élément inséré apparaisse au-dessus de tous les autres.
- **GridPane** : permet de créer une grille d'éléments organisés en lignes et en colonnes.
- **FlowPane** : permet de ranger des éléments de façon à ce qu'ils se positionnent automatiquement en fonction de leur taille et de celle du layout.
- **TilePane** : est similaire au FlowPane, chacune de ses cellules fait la même taille.
- **AnchorPane** : permet de fixer un élément graphique par rapport à un des bords de la fenêtre : top, bottom, right et left.

# Structure d'une application JavaFX

```
import javafx.application.Application;
import javafx.stage.Stage;

public class JavaFxApp1 extends Application {

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        /* Code de l'application ICI */
    }
}
```



# Exemple d'une application JavaFX

```
public class JavaFXApp1 extends Application {  
1   public static void main(String[] args) { launch(args); }  
   @Override  
2   public void start(Stage primaryStage) throws Exception {  
3       //Créer le groupe root BorderPane qui structure le contenu en 5 zones:  
4       //TOP, Bottom, Center, Left et Right  
       BorderPane root=new BorderPane();  
       //créer un label Nom  
       Label labelNom=new Label( text: "Nom :");  
       //créer une zone de texte pour saisir le nom  
       TextField fieldNom=new TextField();  
       //créer deux composants button pour ajouter et pour supprimer une voiture  
       Button buttonAdd=new Button( text: "Ajouter");  
       Button buttonDel=new Button( text: "Supprimer");  
       //Créer le layout HBox qui permet de structurer les éléments horizontalement  
       HBox hbox=new HBox();  
       //définir la marge entre le contenu et les bords du conteneur hbox  
       hbox.setPadding(new Insets( topRightBottomLeft: 15));  
       //définir l'espace des éléments contenus dans hbox  
       hbox.setSpacing(5);  
       //ajouter les composants labelNom, fieldNom, buttonAdd, buttonDel dans hbox  
       hbox.getChildren().addAll(labelNom,fieldNom,buttonAdd,buttonDel);  
       //mettre le hbox dans le centre de BorderPane  
       root.setTop(hbox);  
   }
```

# Exemple d'une application JavaFX

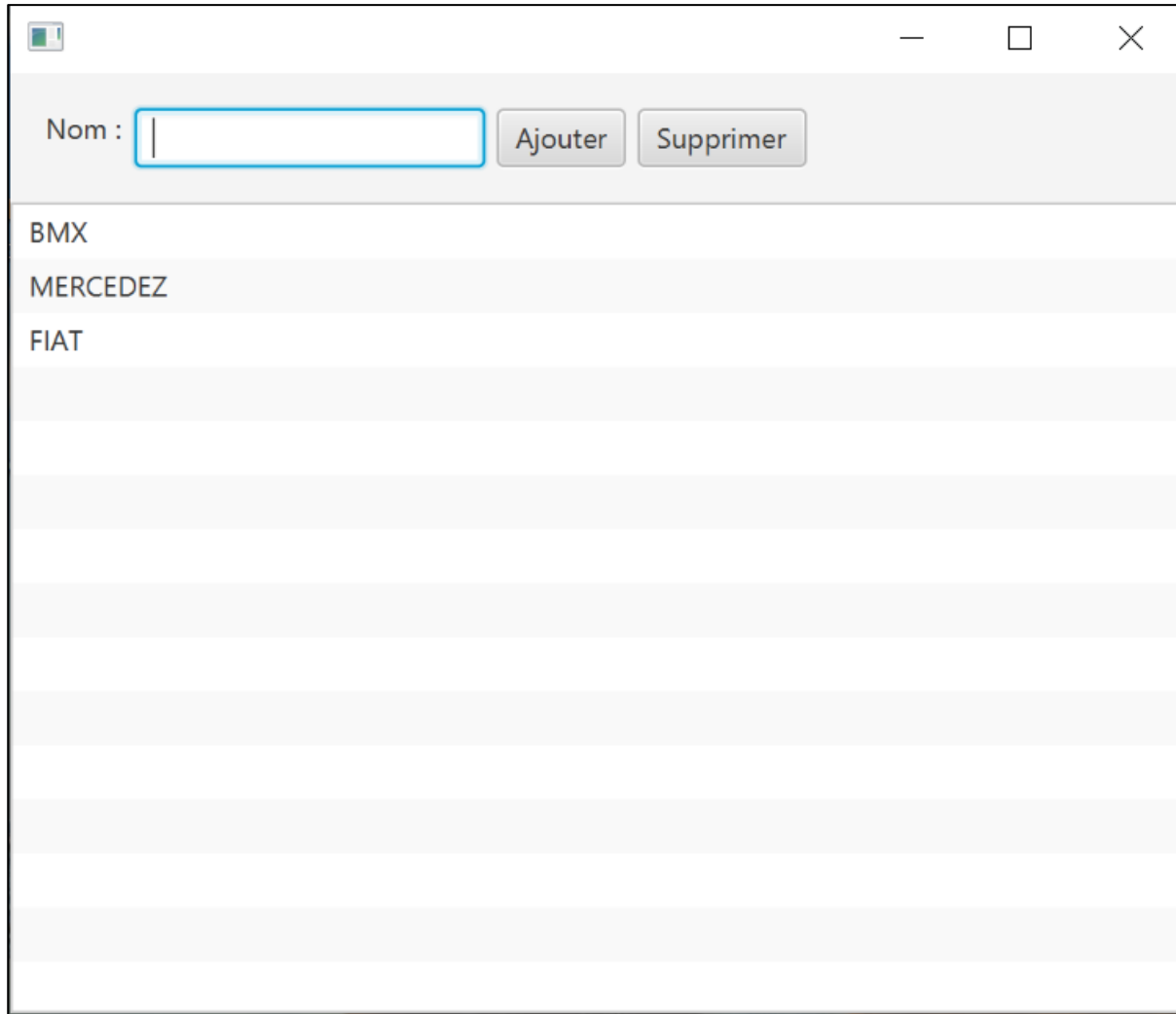
```
//Créer une ListView
ListView<String> listView=new ListView<>();
//ajouter quelque éléments à la listView
listView.getItems().addAll( ...elements: "BMX", "MERCEDEZ", "FIAT");
//mettre la listView dans le centre de BorderPane
root.setCenter(listView);
//Créer le composant scene avec BorderPane comme racine
Scene scene=new Scene(root, width: 500, height: 400);
//afficher la scène dans le stage
primaryStage.setScene(scene);
//afficher le stage
primaryStage.show();

buttonAdd.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        String nom=fieldNom.getText();
        listView.getItems().add(nom);
    }
});
```

# Exemple d'une application JavaFX

```
buttonDel.setOnAction(new EventHandler<ActionEvent>() {  
    @Override  
    public void handle(ActionEvent event) {  
        //récupérer l'indice de l'élément sélectionné  
        int indice=listView.getSelectionModel().getSelectedIndex();  
        if(indice>=0) {  
            listView.getItems().remove(indice);  
        }else{  
            Alert alert=new Alert(Alert.AlertType.WARNING);  
            alert.setContentText("Veuillez sélectionner un élément ");  
            alert.show();  
        }  
    }  
});  
}
```

# Exemple d'une application JavaFX



The image shows a JavaFX application window with a standard title bar (minimize, maximize, close buttons). The window contains a form with a label "Nom :" followed by a text input field. To the right of the input field are two buttons: "Ajouter" and "Supprimer". Below the input field is a list box containing the text "BMX", "MERCEDEZ", and "FIAT". The list box has a light gray background and is surrounded by a thin border. The entire application is enclosed in a blue border.

Nom
BMX
MERCEDEZ
FIAT

## Méthode basée sur FXML

- JavaFX offre la possibilité de déclarer la structure des composants de l'interface graphique dans un fichier XML.
- Le fichier FXML représente la vue de l'application.
- Au démarrage de l'application, le fichier XML est lu, les composants sont instanciés et l'interface graphique s'affiche.
- Pour programmer les réponses aux événements produits dans les composants, on crée un Contrôleur associé à la vue FXML.

# Méthode basée sur FXML

```
<?import javafx.geometry.Insets?>
<BorderPane xmlns="http://javafx.com/javafx"
  xmlns:fx="http://javafx.com/fxml" fx:controller="gui.LayoutController">
  <top>
    <HBox spacing="10">
      <padding>
        <Insets top="10" right="10" left="10" bottom="10"></Insets>
      </padding>
      <children>
        <Label text="Nom : "></Label>
        <TextField fx:id="textFieldNom" style="-fx-background-color: yellow"></TextField>
        <Button text="Ajouter" onAction="#addFruits"></Button>
      </children>
    </HBox>
  </top>
  <center>
    <VBox spacing="10">
      <padding>
        <Insets top="10" right="10" left="10" bottom="10"></Insets>
      </padding>
      <ListView fx:id="listView1">
        <items>
          <FXCollections fx:factory="observableArrayList">
            <String fx:value="BMW"></String>
            <String fx:value="DACIA"></String>
          </FXCollections>
        </items>
      </ListView>
    </VBox>
  </center>
</BorderPane>
```

# Méthode basée sur FXML

- Le contrôleur VoitureController :

```
public class VoitureController implements Initializable{  
    @FXML  
    private TextField fieldMat;  
    @FXML  
    private TextField fieldNom;  
    @FXML  
    private TextField fieldPrix;  
    @FXML  
    private ListView listeView;  
    @FXML  
    private Button buttonAdd;  
    private Connection connx;
```

# Méthode basée sur FXML

- Le contrôleur VoitureController (Suite):

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    try {
        connx= ConnexionDB.getConnection();
        Statement stm=connx.createStatement();
        ResultSet rs=stm.executeQuery( sql: "select * from voitures");
        while (rs.next()){
            listView.getItems().add(new Voiture(rs.getString( columnLabel: "matricule"),rs.getString( columnLabel: "nom")));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void ajouterVoiture(){
    try {
        Statement stm=connx.createStatement();
        String mat=fieldMat.getText();
        String nom=fieldNom.getText();
        float prix=Float.parseFloat(fieldPrix.getText());
        stm.executeUpdate( sql: "insert into voitures vGBalues(null, '"+mat+"', '"+nom+"', '"+prix+"')");
        listView.getItems().add(new Voiture(mat,nom));
    }catch(Exception e){
        e.printStackTrace();
    }
}
```



# Méthode basée sur FXML

- Le code de l'application JAVA :

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;

public class JavaFXApp4 extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage primaryStage) throws Exception {
        BorderPane borderPaneRoot= FXMLLoader.load(getClass().getResource("Layout.fxml"));
        Scene scene=new Scene(borderPaneRoot,400,400);
        scene.getStylesheets().add(getClass().getResource("myStyle.css").toString());
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

# Méthode basée sur FXML

- Style Css:

```
/* styles.css */
.root {
  -fx-font-family: 'Arial';
  -fx-background-color: #f0f0f0;
}

.button {
  -fx-background-color: #4CAF50;
  -fx-text-fill: white;
  -fx-padding: 10px 20px;
  -fx-font-size: 16px;
  -fx-border-radius: 5px;
}

.button:hover {
  -fx-background-color: #45a049;
}

.label {
  -fx-font-size: 18px;
  -fx-text-fill: #333;
}
```