JavaFx

JavaFx application

JavaFx utilise une analogie avec le théâtre pour bâtir la structure de ses applications.

- Le programme **Main** dérive de **Application**
- Le **Stage** (primaryStage) est le théâtre dans lequel les éléments de l'application vont évoluer, il correspond à la fenêtre
- La **Scene** est l'élément qui permettra de faire apparaître ces éléments, elle correspond au décor ou au lieu de l'action
- La Scene contient un élément parent (noeud Root) qui contient lui même d'autres noeuds

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            Scene scene = new Scene(new BorderPane());
scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm
());
            primaryStage.setScene(scene);
            primaryStage.setTitle("First application JavaFx");
            primaryStage.show();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

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

Séparation des couches

Chargement d'une vue à partir d'un fichier fxml :

```
    @Override
    public void start(Stage primaryStage) {
        try {
            BorderPane root =
(BorderPane)FXMLLoader.load(getClass().getResource("Sample.fxml"));
            Scene scene = new Scene(root,400,400);
scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm
());
            primaryStage.setScene(scene);
            primaryStage.show();
        } catch(Exception e) {
```

```
            e.printStackTrace();
        }
    }
```

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

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.MenuButton?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>


<BorderPane xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/9.0.1"
fx:controller="application.SampleController">
   <top>
      <MenuButton mnemonicParsing="false" text="MenuButton"
BorderPane.alignment="CENTER">
        <items>
          <MenuItem mnemonicParsing="false" text="Action 1" />
          <MenuItem mnemonicParsing="false" text="Action 2" />
        </items>
      </MenuButton>
   </top>
   <center>
      <AnchorPane prefHeight="200.0" prefWidth="468.0"
BorderPane.alignment="CENTER">
         <children>
            <Button fx:id="btAdd" layoutX="14.0" layoutY="14.0"
mnemonicParsing="false" onAction="#btAddClick" prefHeight="25.0" prefWidth="115.0"
text="Button" />
         </children>
      </AnchorPane>
   </center>
</BorderPane>
```

Association d'écouteurs

Java 8 permet l'utilisation des Lambda expressions :

```
StackPane elem=new StackPane();
elem.setOnMouseMouve((e)->{system.out.println("x : "+e.getX()+",y : "+e.getY())});
```

Bind sur les propriétés d'un objet

```
    canvas.heightProperty().bind(p.heightProperty());
```

Remplissage et sélection sur un TableView

```
    @FXML
    private void initialize() {
        // Initialize the person table with the two columns.
        prenomColumn.setCellValueFactory((CellDataFeatures<Utilisateur, String>
feature) -> {
            Utilisateur user = feature.getValue();
            return new SimpleObjectProperty<>(user.getPrenom());
        });
        nomColumn.setCellValueFactory((CellDataFeatures<Utilisateur, String>
feature) -> {
            Utilisateur user = feature.getValue();
            return new SimpleObjectProperty<>(user.getNom());
        });
        showUser(null);
personnTable.getSelectionModel().selectedItemProperty().addListener((observable,
oldValue, newValue) -> showUser(newValue));
    }
```

Ressources

jfxtras	http://jfxtras.org/
	Download jar

Articles

- event handling