GUI avec JavaFX 8+

Plan

JavaFX en quelques points-clés

Pour faire quoi ?

- Desktop applications
- RIAs (Rich Internet Applications)

Philosophie

- C'est une API Java $\Rightarrow$ peut utiliser toute API Java
- Sépare apparence/style et code métier $\Rightarrow$ FXML + Java

Disponibilité

- Intégrée aux JDKs 8 à 10. Module indépendant depuis Java 11
- APIs séparées ▸ JavaFX 8 API ▸ JavaFX 17 API

Principales caractéristiques de JavaFX

Basée sur des librairies éprouvées

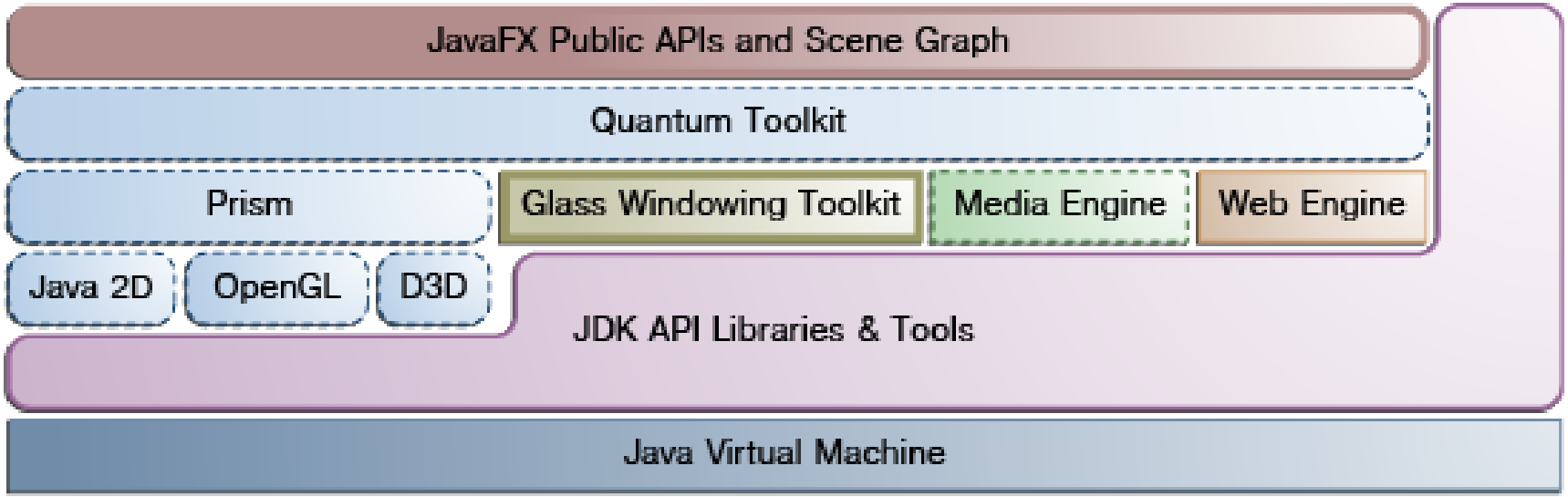
- **Multimédia** : utilise le framework GStreamer
- **WebView** : basé sur WebKitHTML $\Rightarrow$ JS, HTML 5, etc.
- **Rendu graphique** : moteur PRISM (fonctions GPU avancées)

Principales caractéristiques de JavaFX

Facilités

- *single JavaFX application thread approach*
- Nombreux composants d'UI + CSS optionnel pour le style
- MVC simplifié (*observable lists and maps*)
- 3D natif (contrairement à Swing)
- Canvas API ($\approx$ Swing) : fonctions de dessin
- Printing / Hi-DPI / Rich text / Multitouch / déploiement autonome
- Interopérable avec Swing
- Disponibilité d'un outil WYSIWYG : *Scene Builder* ▸ plus d'information

Architecture JavaFX



Scene Graph : arbre de noeuds représentant la hiérarchie des éléments visuels

Exemples de `javafx.scene.control`

▸ plus d'information

Exemples de *javafx.scene.layout*

Gestionnaires de mise en page des noeuds

- **BorderPane** : top, bottom, right, left, or center region.
- **HBox** arranges nodes horizontally in a single row.
- **VBox** arranges nodes vertically in a single column.
- **StackPane** : nodes in a back-to-front single stack.
- **GridPane** : grid of rows and columns in which to lay out nodes.
- **FlowPane** : nodes in either a horizontal or vertical ”flow,” using specified boundaries.
- **TilePane** : nodes in uniformly sized layout cells or tiles
- **AnchorPane** : create anchor nodes to the top, bottom, left side, or center of the layout.

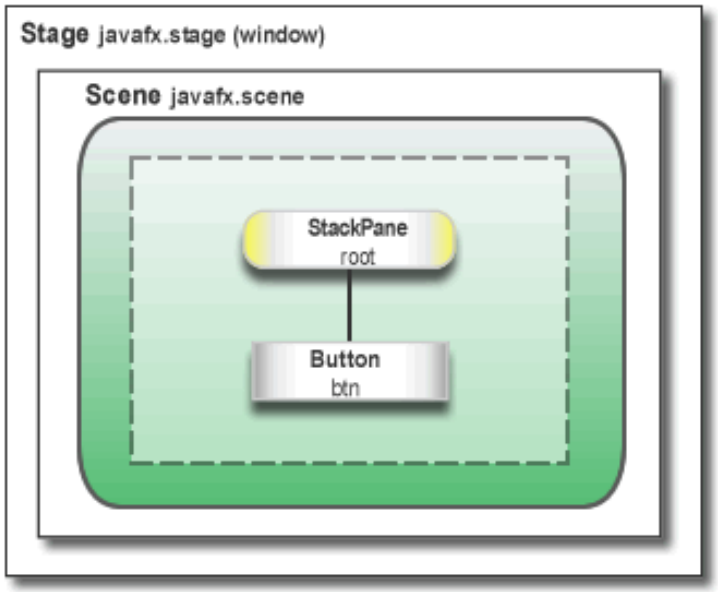
▸ plus d'information

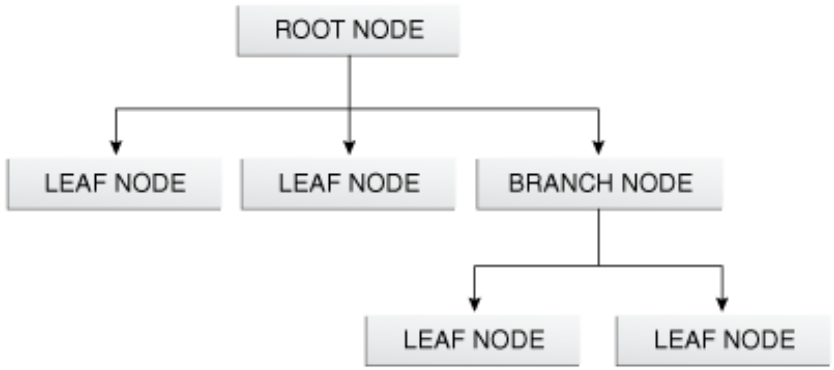
Code source pour Hello World

HelloWorldFX.java

```
public class HelloWorldFX extends javafx.application.Application {
  @Override
  public void start(Stage primaryStage) { // main entry point
    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 node, resizable wrt. the scene
    root.getChildren().add(btn);

    Scene scene = new Scene(root, 300, 250); // container for all content (window)

    primaryStage.setTitle("Hello World!");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

  // not necessarily required, i.e. if the app is created by the JavaFX Packager tool
  public static void main(String[] args) {
    launch(args);
  }
}
```

Scene graph pour Hello World





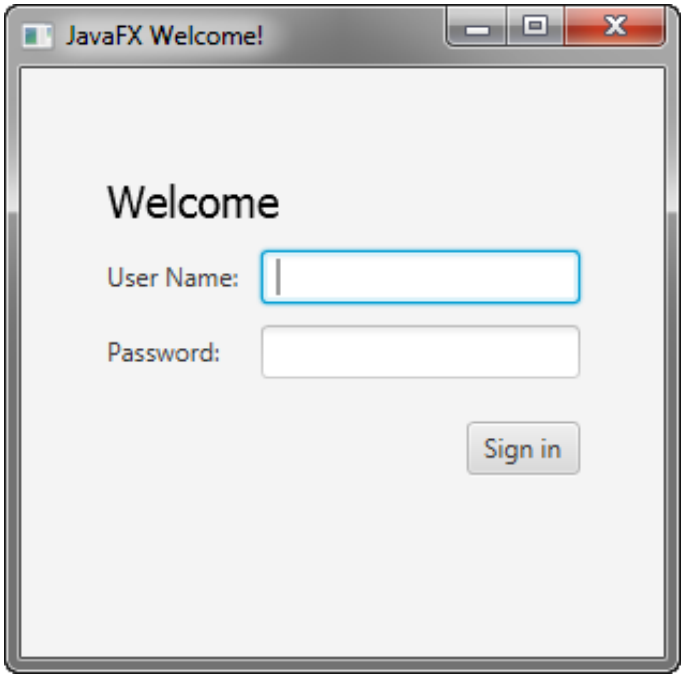
▸ plus d'information

Un formulaire

Login.java

```
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.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
    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, 1, 6);
    btn.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent e) {
            actiontarget.setFill(Color.FIREBRICK);
            actiontarget.setText("Sign in button pressed");
        }
    });
    Scene scene = new Scene(grid, 300, 275);
    primaryStage.setScene(scene);          primaryStage.show();
}
```

Rendu



▸ plus d'information

Styles avec CSS

Principes

- La décoration d'une *scene* sera définie par un fichier CSS
- $\Rightarrow$ enlève les lignes de mise en forme du code Java

Ajout d'un style à une *scene*

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

▸ plus d'information

Définition de styles globaux pour la *scene*

`Login.CSS` affecte tous ces composants

```
.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 );
}
.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);
}
```

Styles pour des composants identifiés

`Login.java` : définition des id des composants

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

`Login.CSS` styles des composants identifiés

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

Rendu

▸ plus d'information

Création d'UI avec FXML

Principes

- L'ensemble des composants d'une *scene* sera défini en FXML
- $\Rightarrow$ enlève les éléments d'UI du code Java

Chargement du graphe de composants de la *scene*

```
Parent root = FXMLLoader.load(
getClass().getResource("fxml_example.fxml") );
Scene scene = new Scene(root, 300, 275);
```

▸ plus d'information

Définition de la *scene* en FXML 1/3

`fxml_example.fxml` en-tête et import

```
<?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.*?>
```

Définition de la *scene* en FXML 2/3

`fxml_example.fxml` définition de la *scene*

```
<GridPane fx:controller="fxml.FXMLExampleController"
    xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">

  <padding><Insets top="25" right="25" bottom="10" left="25"/></padding>

  <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"/>

. . .
```

Définition de la *scene* en FXML 3/3

`fxml_example.fxml` définition de la *scene*

```
. . .
    <HBox spacing="10" alignment="bottom_right"
        GridPane.columnIndex="1" GridPane.rowIndex="4">
        <Button text="Sign In"
        onAction="#handleSubmitButtonAction"/>//trigger controller's method
  </HBox>

  <Text
    fx:id="actiontarget"
    GridPane.columnIndex="0"
    GridPane.columnSpan="2"
    GridPane.halignment="RIGHT"
    GridPane.rowIndex="6"/>


</GridPane>
```

Définition du contrôleur pour les événements

FXMLExampleController.java

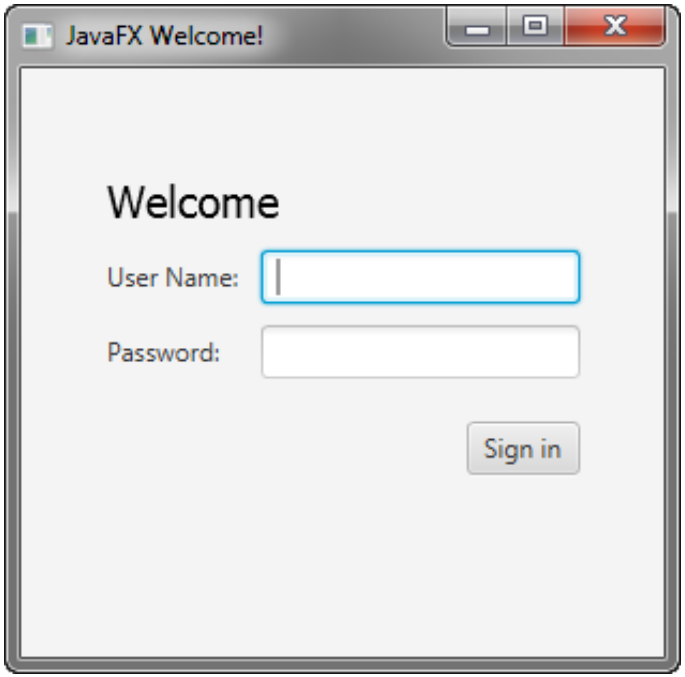
```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.text.Text;

public class FXMLExampleController {
  //annotation used to tag nonpublic member fields and handler methods
  //for use by FXML markup
  @FXML
  private Text actiontarget;

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

}
```

Chargement du graphe de composants de la *scene*

FXMLExample.java

```
public class FXMLExample extends Application {

  @Override
    public void start(Stage stage) throws Exception {
       Parent root = FXMLLoader.load(getClass().getResource("fxml_example.fxml"));

        Scene scene = new Scene(root, 300, 275);

        stage.setTitle("FXML Welcome");
        stage.setScene(scene);
        stage.show();
    }

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

Rendu : il manque la CSS



Ajout de la CSS dans le FXML

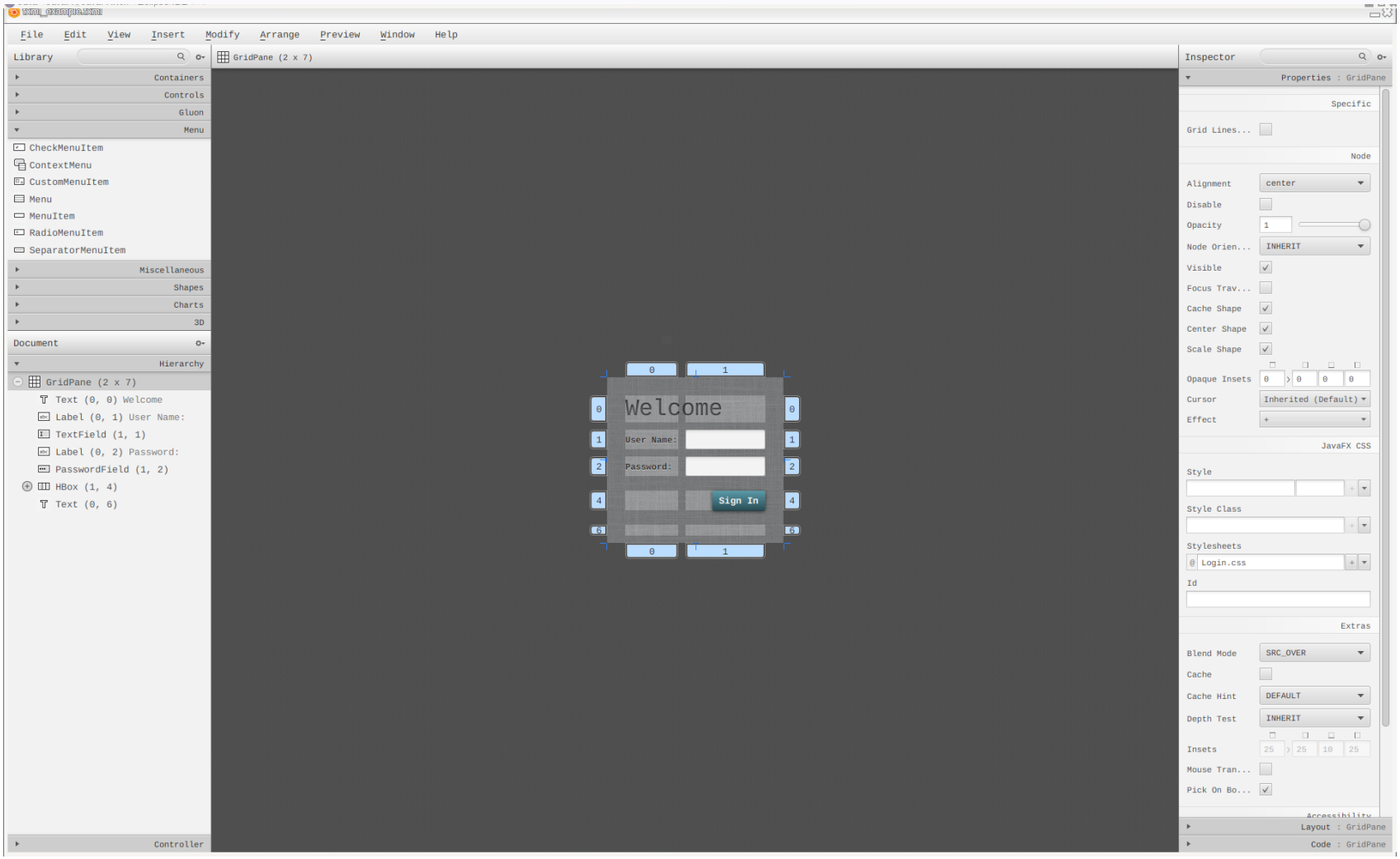
`fxml_example.fxml` : style / id / css loading

```
. . .
<GridPane fx:controller="fxmlexample.FXMLExampleController"
    xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10"
    styleClass="root">
. . .
<Text id="welcome-text" text="Welcome"
        GridPane.columnIndex="0" GridPane.rowIndex="0"
        GridPane.columnSpan="2"/>

. . .
  <stylesheets>
      <URL value="@Login.css" />
  </stylesheets>

/*The @ symbol before the name of the style sheet in the URL
indicates that the style sheet is in the same directory as the FXML file.*/

</GridPane>
```

Rendu

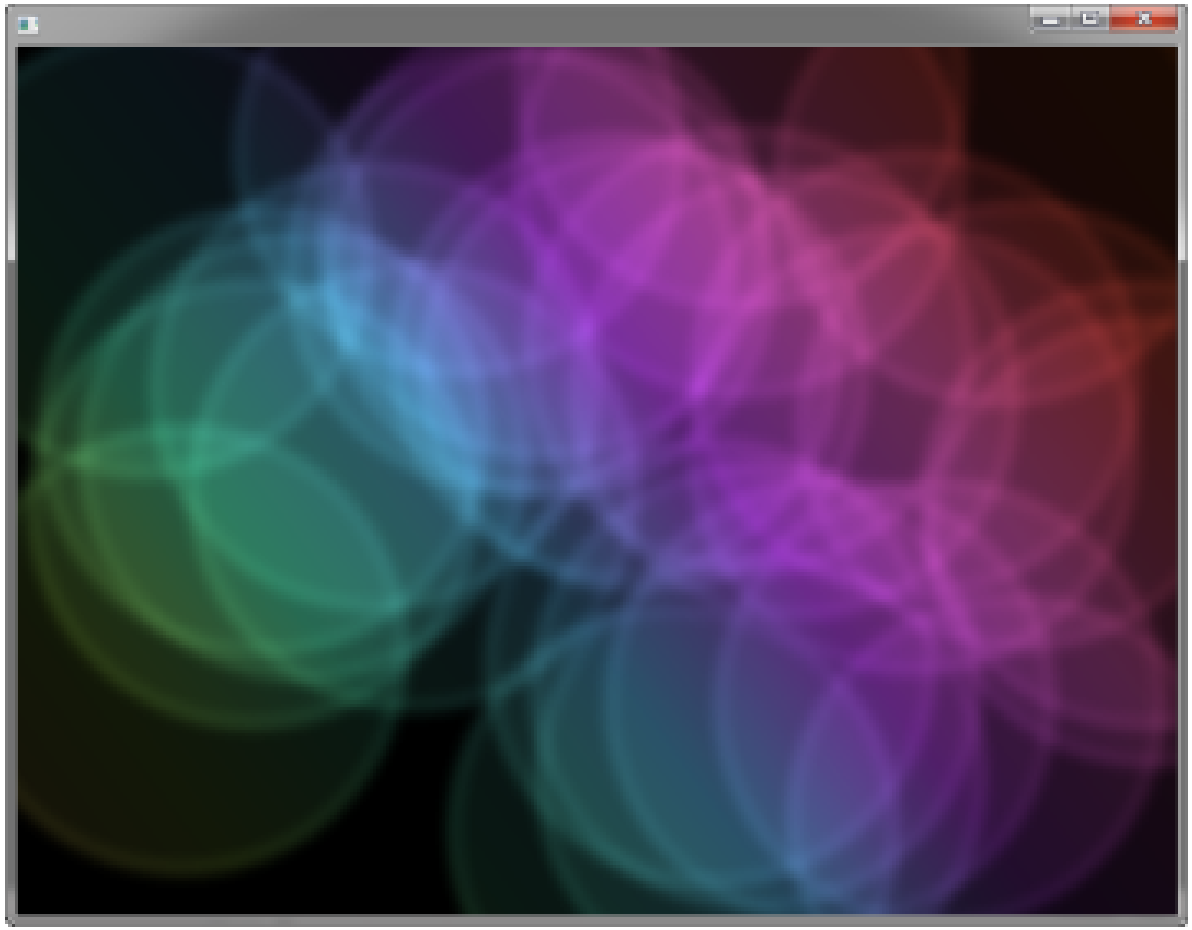

JavaFX Welcome
Welcome
User Name:
Password:
Sign in
Sign in button pressed

JavaFX Scene Builder

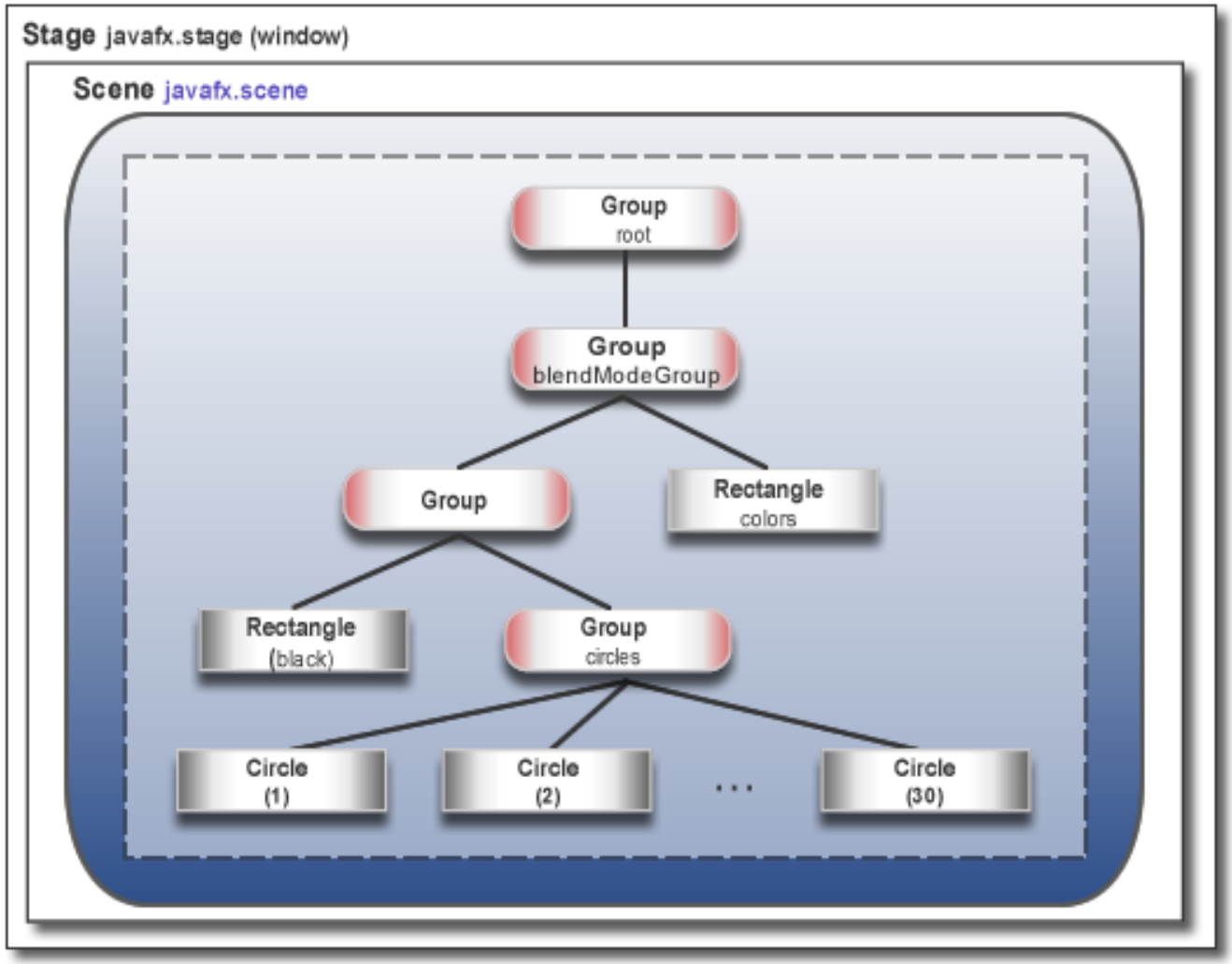


L'application *Colorful Circles*

▸ ColorfulCircles.zip



Graphe de l'application Colorful Circles



Animation avec `KeyFrame` et `KeyValue`

Extrait de *start()* de `ColorfulCircles.java`

```
Timeline timeline = new Timeline();
for (Node circle: circles.getChildren()) {
    timeline.getKeyFrames().addAll(
        new KeyFrame(Duration.ZERO, // set start position at 0
            new KeyValue(circle.translateXProperty(), random() * 800),
            new KeyValue(circle.translateYProperty(), random() * 600)
        ),
        new KeyFrame(new Duration(40000), // set end position at 40s
            new KeyValue(circle.translateXProperty(), random() * 800),
            new KeyValue(circle.translateYProperty(), random() * 600)
        )
    );
}
// play 40s of animation
timeline.play();
```

Dessins personnalisés

javafx.scene.canvas.Canvas

- Noeud correspondant à une "feuille blanche" pour le dessin :

```
Canvas c = new Canvas();
c.setHeight(512);
c.setWidth(512);
GraphicsContext gc = c.getGraphicsContext2D();
```

javafx.scene.canvas.GraphicsContext

- Objet "pinceau", utilisé pour dessiner, e.g. sur un Canvas

```
gc.setFill(Color.valueOf("ff0000"));//remplissage
gc.fillRect(100, 100, 200, 200);
gc.setStroke(Color.valueOf("0000ff"));//ligne
gc.strokeRect(200, 200, 200, 200);
```

Aller plus loin

Tuto et exemples de code

- ▸ JavaFX ensemble 8 : 100 exemples de projets illustratifs ▸ Jakob Jenkov's JavaFX Tutorial
- ▸ 20 exemples d'applications professionnelles réalisées avec JavaFx

Quelques librairies et outils additionnels

- Plus de *controls* et de fonctionnalités : ▸ ControlsFX ▸ JFXtras
- SceneBuilder : ▸ Gluon

Liste exhaustive de ressources

- ▸ Awesome JavaFX

Ce cours est basé sur le tutoriel JavaFX par Oracle :

▸ Oracle JavaFX Tutorial