

Java

JavaFX

Overview

- Java GUI
 - Java 1.0 (1996) – AWT
 - using native GUI components
 - Java 1.2 (2000) – Swing
 - GUI completely in Java
 - JavaFX (2007)
 - new technology
 - running on the Java VM
 - but own language
 - declarative
 - intended as a competitor to Flash
 - failed
 - JavaFX 2.0 (2011)
 - only API (own language abandoned)
 - since JDK 7 update 6 a part of std library (JavaFX 2.2)
 - Java 8 – JavaFX 8
 - no further development of Swing
 - Java 11 – JavaFX decoupled from std library

Hello World

```
public class HelloWorld extends Application {  
  
    @Override  
    public void start(Stage stage) {  
        Label message = new Label("Hello, JavaFX!");  
        message.setFont(new Font(100));  
        stage.setScene(new Scene(message));  
        stage.setTitle("Hello");  
        stage.show();  
    }  
  
    // no main()  
}
```



no J prefix

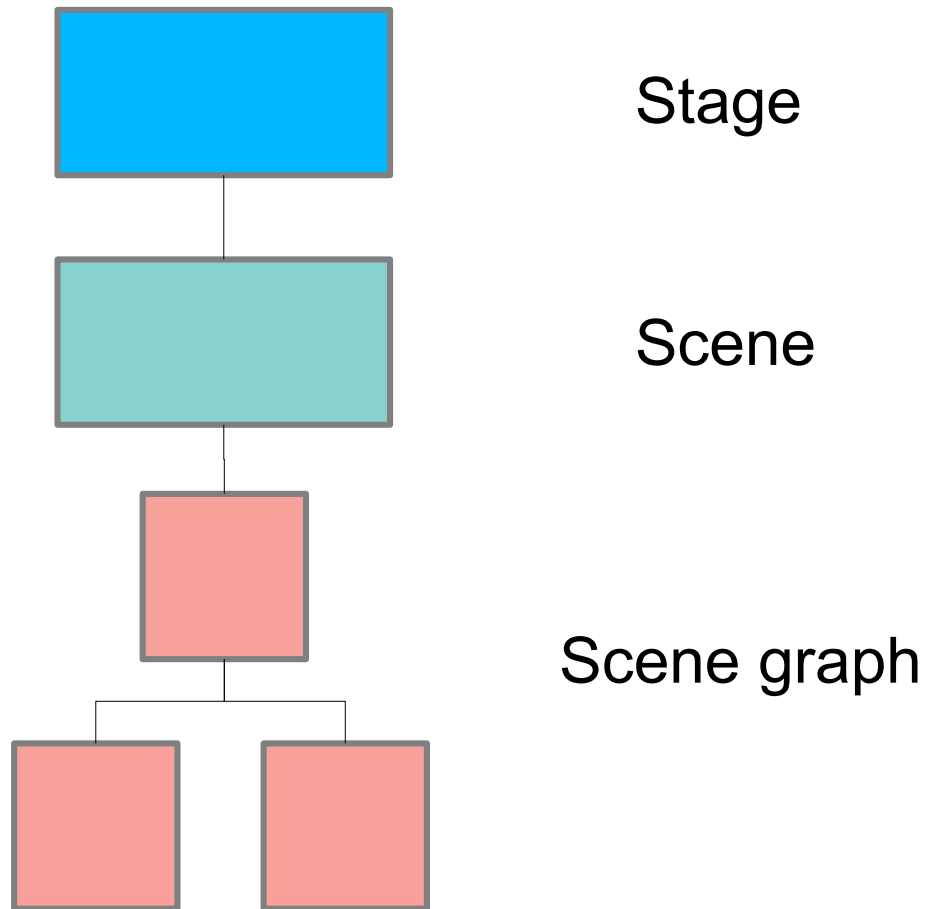


Hello World

- explicit main() necessary in older version

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

Application structure



Events

- similar as in Swing/AWT
- BUT typically – listening on property changes
 - i.e. not listening directly on components
 - there are exceptions, e.g. buttons

```
Slider slider = new Slider(10, 100, 100);  
Label message = new Label("Hello, JavaFX!");
```

```
slider.valueProperty().addListener(property ->  
message.setFont(new Font(slider.getValue())));
```

Properties of components

- interface Property<T>
 - void addListener(InvalidationListener listener)
 - void addListener(ChangeListener<? super T> listener)
 - void bind(ObservableValue<? extends T> observable)
 - void bindBidirectional(Property<T> other)
 - ...
- implementace
 - class ObjectProperty<T>
 - class IntegerProperty
 - class BooleanProperty
 - class StringProperty
 - ...

Properties – implementation ex.

```
private StringProperty text =  
    new SimpleStringProperty("");  
  
public final StringProperty textProperty() {  
    return text;  
}  
  
public final void setText(String newValue){  
    text.set(newValue);  
}  
  
public final String getText() {  
    return text.get();  
}
```


Properties – listeners

- `InvalidationListener`
 - called if the current property value is not valid anymore
 - allows for “lazy” evaluation

```
void invalidated(Observable observable)
```

- `ChangeListener`
 - called if the current property value has changed
 - it is necessary to evaluate the new value
 - does not allow for “lazy” evaluation

```
void changed(ObservableValue<? extends T>  
             observable, T oldValue, T newValue)
```

Properties – binding

- automated updating of a property when another one is changed
 - internally implemented via listeners

```
text1.textProperty().bind(text2.textProperty());
```

```
text1.textProperty().bindBidirectional(  
    text2.textProperty());
```

- class Bindings
 - static methods for easy creation of bindings

Layout

- creating layout of components
 - JavaFX Scene Builder
 - GUI build visually by “clicking”
 - “manually”
 - creating instances of components
 - definition in FXML
- similar as in Swing
 - panes instead of layout managers

Layout

- BorderPane (as BorderLayout)
- HBox
- VBox
- GridPane (as GridBagLayout)
- TilePane (as GridLayout)
- ...

```
VBox pane = new VBox();  
pane.getChildren().addAll(button1, button2,);  
pane.setPadding(new Insets(10));
```

FXML

```
<?import java.net.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<GridPane
    fx:controller="fxmlexample.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"/>
    ...
```

FXML

- usage

```
public void start(Stage stage) {
    try {
        Parent root = FXMLLoader.load(getClass().
                                     getResource("file.fxml"));
        stage.setScene(new Scene(root));
        stage.show();
    } catch (IOException ex) {
        ex.printStackTrace();
        System.exit(0);
    }
}
```

FXML

- adding listeners/bindings

a)

```
<TextField id="username" ...
```

```
TextField username =  
    (TextField) root.lookup("#username");
```

b)

```
<GridPane xmlns:fx="http://javafx.com/fxml" hgap="10"  
vgap="10" fx:controller="LoginDialogController">  
<Button fx:id="okButton" text="Ok" />
```

```
public class LoginDialogController implements  
Initializable {  
    @FXML private TextField username;  
    @FXML private Button okButton;  
    public void initialize(URL url, ResourceBundle rb){  
        okButton.disableProperty().bind(...
```

CSS

- visual appearance of UI can be set via CSS

```
Scene scene = new Scene(pane);  
scene.getStylesheets().add("scene.css");
```

- JavaFX specific attributes are used

```
#pane {  
    -fx-padding: 0.5em;  
    -fx-hgap: 0.5em;  
    -fx-vgap: 0.5em;  
    -fx-background-image: url("bg.jpg")  
}
```


Animace apod.

- ScaleTransition

```
ScaleTransition st =  
    new ScaleTransition(Duration.millis(3000));  
st.setByX(1.5);  
st.setByY(1.5);  
st.setNode(button);  
st.setCycleCount(Animation.INDEFINITE);  
st.setAutoReverse(true);  
st.play();
```

- FadeTransition
- RotateTransition
- DropShadow
- ...

Component overview

- Accordion
- Button
- CheckBox
- ChoiceBox
- ColorPicker
- ComboBox
- DatePicker
- Label
- ListView
- Menu
- MenuBar
- PasswordField
- ProgressBar
- RadioButton
- Slider
- Spinner
- SplitMenuButton
- SplitPane
- TableView
- TabPane
- TextArea
- TextField
- TitledPane
- ToggleButton
- ToolBar
- TreeTableView
- TreeView

Charts

- BarChart
- BubbleChart
- LineChart
- PieChart
- ScatterChart
- StackedAreaChart
- StackedBarChart

Other

- WebView
 - rendering HTML
 - based on WebKit
- playing video
- ...

Threads

- similarly as in Swing
 - single-threaded UI
- UI thread
 - all UI changes from this thread

Platform.runLater(Runnable runnable)

 - long-running operations -> own thread
- javafx.concurrent
 - Worker interface
 - similar to the SwingWorker class
 - Task, Service, ScheduledService
 - implementations of the Worker interface
 - Service executes Tasks

JavaFX Task<V>

- implements
`java.util.concurrent.FutureTask<T>`
- executing
 - Thread th = new Thread(task);
th.setDaemon(true);
th.start();
 - or
`ExecutorService.submit(task);`
- one-time action
 - cannot be reused

JavaFX Task<V>

- property
 - ReadOnlyObjectProperty<Throwable> **exception**
 - ReadOnlyStringProperty **message**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onCancelled**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onFailed**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onRunning**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onScheduled**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onSucceeded**
 - ReadOnlyDoubleProperty **progress**
 - ReadOnlyBooleanProperty **running**
 - ReadOnlyObjectProperty<Worker.State> **state**
 - ReadOnlyStringProperty **title**
 - ReadOnlyDoubleProperty **totalWork**
 - ReadOnlyObjectProperty<V> **value**
 - ReadOnlyDoubleProperty **workDone**

JavaFX Service<V>

- non-visual component
- executes tasks
 - uses Executor (by default ThreadPoolExecutor)

```
class MyService extends Service<String> {  
    protected Task createTask() {  
        return new Task<String>() {  
            ...  
        };  
    }  
}
```

```
new MyService().start();
```

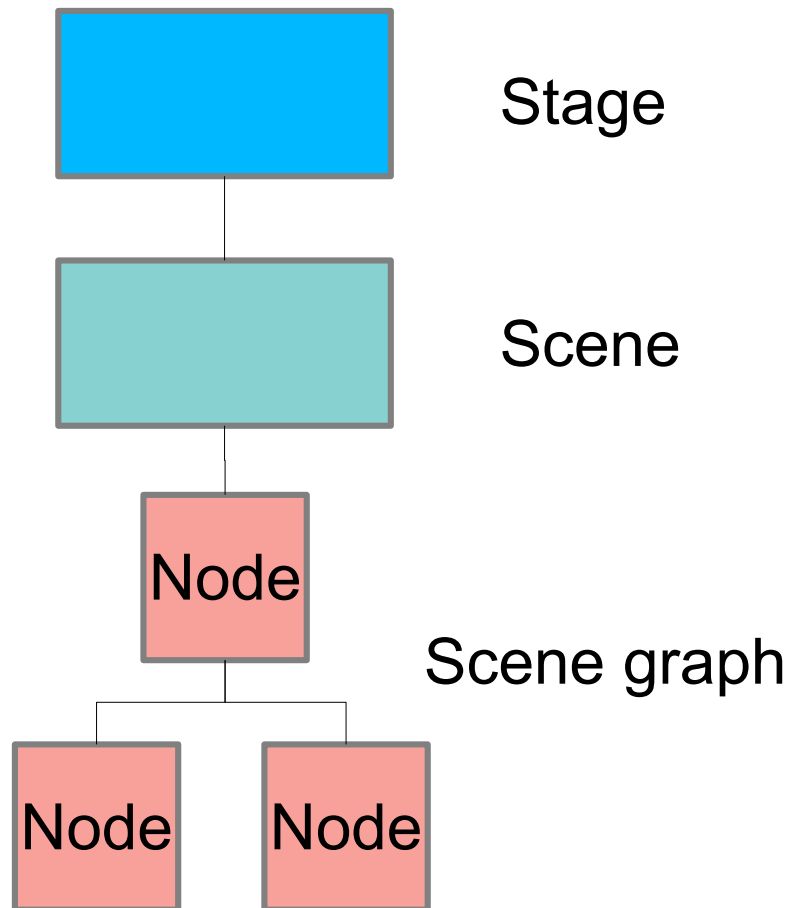

JavaFX Service<V>

- vlastnosti
 - ReadOnlyObjectProperty<Throwable> **exception**
 - ObjectProperty<Executor> **executor**
 - ReadOnlyStringProperty **message**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onCancelled**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onFailed**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onReady**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onRunning**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onScheduled**
 - ObjectProperty<EventHandler<WorkerStateEvent>> **onSucceeded**
 - ReadOnlyDoubleProperty **progress**
 - ReadOnlyBooleanProperty **running**
 - ReadOnlyObjectProperty<Worker.State> **state**
 - ReadOnlyStringProperty **title**
 - ReadOnlyDoubleProperty **totalWork**
 - ReadOnlyObjectProperty<V> **value**
 - ReadOnlyDoubleProperty **workDone**

ScheduledService<V>

- automatically restarts itself after an execution

Application



- Nodes
 - Button
 - Label
 - ...
 - Line
 - Circle
 - Rectangle
 - ...

Canvas

- an image that can be drawn on using a set of graphics commands

```
Canvas canvas = new Canvas(250,250);  
GraphicsContext gc = canvas.getGraphicsContext2D();
```

```
gc.setFill(Color.BLUE);  
gc.fillRect(75,75,100,100);
```

Games

- Canvas + AnimationTimer

SceneBuilder

- a visual tool for UI creation
 - Starting with Oracle Java SE 8u40, Oracle does not provide a separate set of accompanying JavaFX Scene Builder binaries...
- download
 - <http://gluonhq.com/products/scene-builder/>

JFXPanel

- `javafx.embed.swing.JFXPanel`
- Swing component
- content of the component – JavaFX
- warning – „event-dispatching“ threads
 - Swing and JavaFX do not share the thread



Slides version AJ07.en.2019.01

This slides are licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/).