

# Java

## JavaFX

# Přehled

- Java GUI
  - Java 1.0 (1996) – AWT
    - použití nativních GUI komponent
  - Java 1.2 (2000) – Swing
    - kompletní GUI v Javě
  - JavaFX (2007)
    - nová technologie
    - běžící nad Java VM
    - ale vlastní jazyk
      - deklarativní
    - zamýšleno jako konkurence pro Flash
    - neujalo se
  - JavaFX 2.0 (2011)
    - zůstalo pouze API (jazyk zahozen)
  - od JDK 7 update 6 součást std knihovny (JavaFX 2.2)
  - Java 8 – JavaFX 8
  - JDK 11 – JavaFX odstaněna z std knihovny JDK

# 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();  
    }  
  
    // žádný main()  
}
```

žádný J prefix

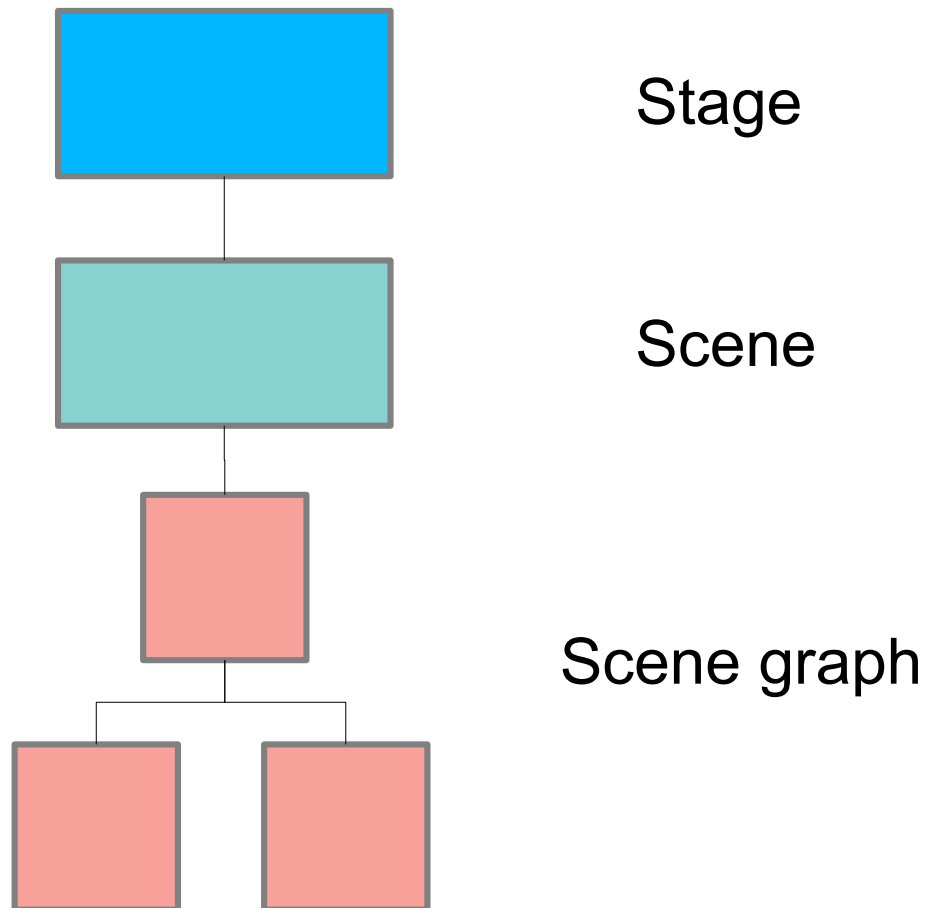


# Hello World

- ve starších verzích bylo potřeba main() přidat explicitně

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

# Struktura aplikace



# Události

- podobně jako ve Swingu/AWT
- ALE typicky se poslouchá na změny vlastností
  - tj. neposlouchá se přímo na komponentě
    - existují výjimky, např. tlačítko

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

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

# Vlastnosti komponent (properties)

- 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
  - ...

# Vlastnosti – příklad implementace

```
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();
}
```



# Vlastnosti – listeners

- `InvalidationListener`
  - volá se pokud současná hodnota vlastnosti přestala platit
  - umožňuje „líné“ vyhodnocení

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

- `ChangeListener`
  - volá se při změně hodnoty vlastnosti
  - je potřeba spočítat i novou hodnotu
  - tj. neumožňuje „líné“ vyhodnocení

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

# Vlastnosti – propojování

- „binding“
- automatické aktualizování vlastnosti při změně jiné vlastnosti
  - interně implementováno pomocí listenerů

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

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

- třída Bindings
  - statické metody pro snadné vytváření propojení

# Layout

- vytvoření rozložení (layout) komponent
  - JavaFX Scene Builder
    - „naklikání“ GUI
  - „ručně“
    - vytváření instancí komponent
  - definice v FXML
- podobné jako ve Swingu
  - panely (panes) místo layout managerů

# Layout

- BorderPane (jako BorderLayout)
- HBox
- VBox
- GridPane (jako GridBagLayout)
- TilePane (jako 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

- použití

```
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

- přidání listenerů/propojení

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

- vzhled UI lze nastavit pomocí CSS

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

- používají se JavaFX specifické atributy

```
#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
- ...

# Přehled komponent

- 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

# Grafy (charts)

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

# Další

- WebView
  - zobrazení HTML
  - založeno na WebKit
- přehrávání videa
- ...

# Vlákna

- podobně jako ve Swingu
  - jednovláknové UI
- UI vlákno
  - všechny změny v UI z tohoto vlákna

Platform.runLater(Runnable runnable)

  - dlouhotrvající operace -> vlastní vlákno
- javafx.concurrent
  - Worker interface
    - obdoba SwingWorker třídy
  - Task, Service, ScheduledService
    - implementace Worker interfacu
    - Service spouští Tasky

# JavaFX Task<V>

- implementuje  
`java.util.concurrent.FutureTask<T>`
- spouštění
  - Thread th = new Thread(task);  
th.setDaemon(true);  
th.start();
  - nebo  
ExecutorService.submit(task);
- jednorázová akce
  - nelze použít opakovaně

# JavaFX Task<V>

- vlastnosti
  - 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>

- komponenta bez zobrazení
- spouští akce (Task)
  - používá Executor (implicitně ThreadPoolExecutor)

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

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



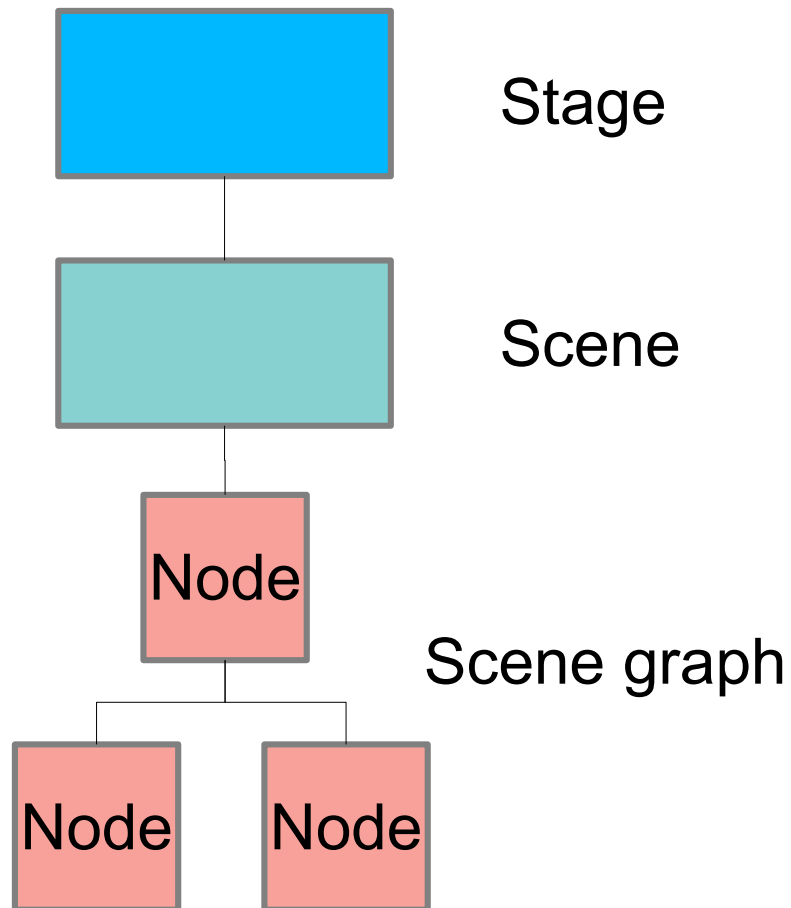
# JavaFX Service<V>

- 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>

- po ukončení se automaticky restartuje

# Aplikace



- Uzly (nodes)
  - Button
  - Label
  - ...
  - Line
  - Circle
  - Rectangle
  - ...

# Canvas

- obrázek, do kterého lze kreslit

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

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

# Hry

- Canvas + AnimationTimer

# SceneBuilder

- vizuální nástroj na tvorbu UI
  - Starting with Oracle Java SE 8u40, Oracle does not provide a separate set of accompanying JavaFX Scene Builder binaries...
- stažení
  - <http://gluonhq.com/products/scene-builder/>

# JFXPanel

- `javafx.embed.swing.JFXPanel`
- Swingová komponenta
- obsah komponenty – JavaFX
- pozor na „event-dispatching“ vlákna
  - Swing a JavaFX vlákno nesdílejí



Verze prezentace AJ07.cz.2019.01

Tato prezentace podléhá licenci [Creative Commons Uveďte autora-Neužívejte komerčně 4.0 Mezinárodní License](https://creativecommons.org/licenses/by-nc/4.0/).