Тема 12 - JavaFx

03 юли 2018 г. 21:49

JavaFx е най-новата библиотека на Oracle за визуални десктоп, както и уеб приложения за разлика от Swing, която се използва само за десктоп приложения.
Някои разлики между Swing и JavaFx:

Swing	JavaFx
Класа, на който седи интерфейса е JFrame на него може да добавяте различни JPanel- и	Класа , на който седи интерфейса е Stage, другите елементи се добавят към Scene клас. В едно приложение може да има повече от една сцена, но може да се показва само една от тях. В тази сцена се намира Scene Graph (граф на сцената), в който се съдържат отделните Node елементи, като Layout, Controls, Shapes
Layout manager-ите са свързани с JPanel-ите, ако искате да добавяте повече от един Layout трябва да създавате допълнителни панели.	Layout-ите са подкласове на Node ,всеки от тях може да съдържа набор от Node-ове. В Layout Node може да се съдържат други Node-ове, като бутони, комбо боксове, чек боксове и др.(Controls), както и други Layout-и. Което прави схемата за построяване на приложението много по-удобна от тази на Swing с неговата асоциация с панели.
Подържа Event-и за управление на входа от потребителя	Event-ите са по-постоянни в тяхното значение и са силно свързани с Properties.
Не поддържа форматиране с CSS	Поддържа форматиране с помощта на CSS, като може да се променя почти всичко по интерфейса чрез него.
Не поддържа touch устройства.	Поддържа touch event-и , и с тях се работи по същият начин като с другите event-и.

Най-добрият начин за запознаването с JavaFx е да създадем приложение, което да покаже пример за това как работи библиотеката.

Преди да преминете нататък прочетете Наръчник за инсталация на JavaFX в Eclipse.pdf

Hello World приложение:

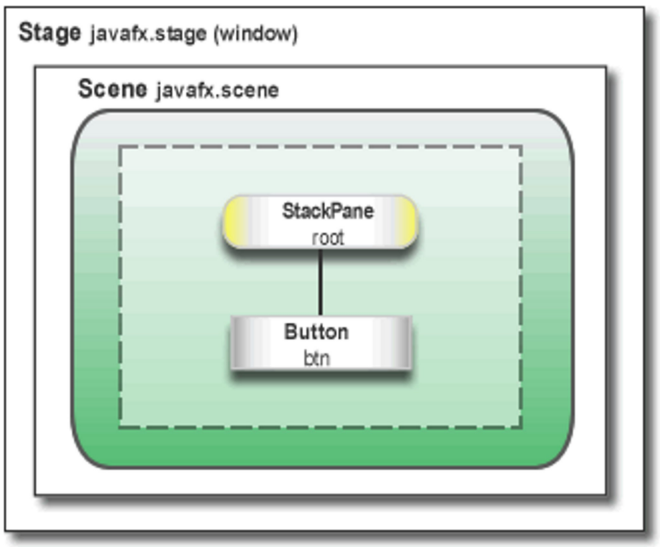
```
//Файл:HelloWorld.java
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloWorld extends Application {

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

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button(); // създаване на бутон
        btn.setText("Say 'Hello World'"); // задаване на текста на бутона
        btn.setOnAction(new EventHandler<ActionEvent>() { // добавяне на
функционално на бутона
```

```
            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

        StackPane root = new StackPane(); // създаване на StackPane Layout
        root.getChildren().add(btn); // добавяне на бутона

        Scene scene = new Scene(root, 300, 250); // създаване на сцената с
StackPane за Layout и размреи 300 широчина 250 височина

        primaryStage.setTitle("Hello World!"); // задавяне на заглавие на
прозореца
        primaryStage.setScene(scene); // задаване на сцената на прозореца
        primaryStage.show(); // показване на прозореца
    }
}
```

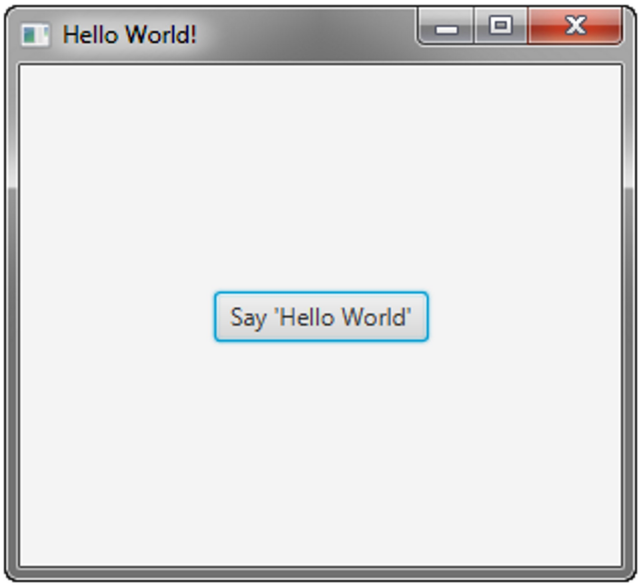
Важните неща за структурата на тази Hello World програма:

- Главният клас за JavaFx приложение наследява класа Application и неговата точка на изпълнение започва от start() метода.
- Получаваме обекта си за Stage, който е главният клас на програмата от метода start(), но трябва да си построим обекта от тип Scene, като в случая му подаваме StackPane за Layout и размерът на прозореца, който искаме.
- Най-долният Node на Scene graph-а е Button, който има EventHandler обявен inline(вграден клас в кода), който принтира съобщение на конзолата.

Графът на сцената изглежда по следният начин:



Програмата изглежда така:



Login Форма :

Създаваме JavaFx проект. След това премахнете кода, който имате за start() метода и добавете този код:

```
@Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("JavaFX Welcome"); // задава заглавие за
прозореца

        primaryStage.show(); // показва прозореца винаги най-отдолу на
метода start()
    }
```

Създаваме GridPane Layout - за разлика от Swing тук няма нужда да създаваме допълнителни класов към които да го добавяме.

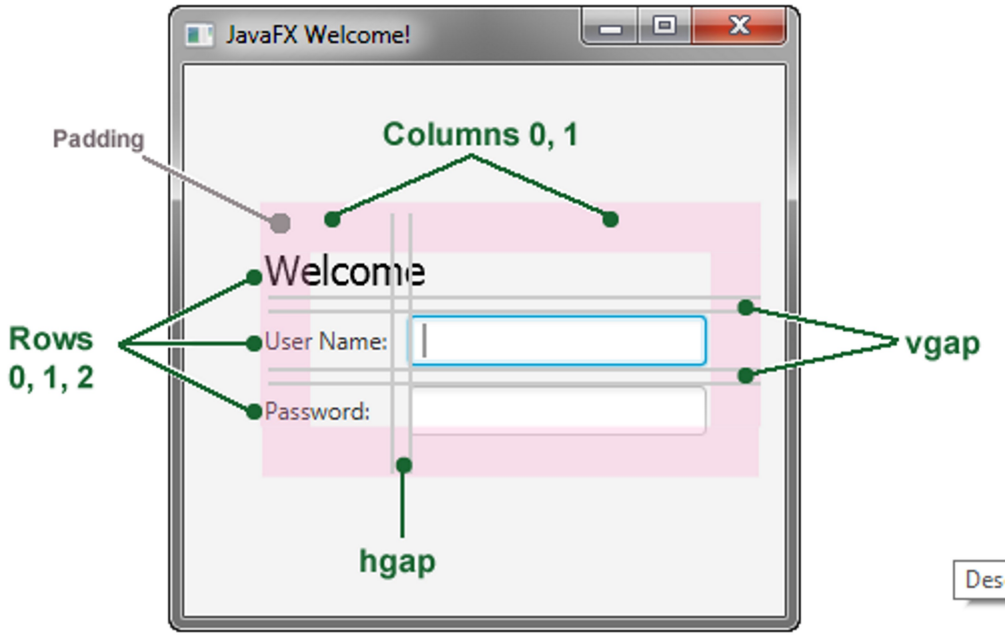
```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER); // променя позицията на Layouta да бъде в
центъра
grid.setHgap(10); // задава разстояние между елементите по хоризонтала
grid.setVgap(10); // задава разстояние между елементите по вертикала
grid.setPadding(new Insets(25, 25, 25, 25)); // добавя разстояние по
краищата на GridPane-а

Scene scene = new Scene(grid, 300, 275); // Създава сцена с размери и задава
GridLayout-а като layout за сцената
primaryStage.setScene(scene); // поставя сцената на прозореца
```

Добавяне на Text, Label, Text Field

```
Text scenetitle = new Text("Welcome"); // създаване на обект от тип Text
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20)); // залагане
на неговият шрифт
grid.add(scenetitle, 0, 0, 2, 1); // добавяне на текста в GridPane, на
позиция 0 колона 0 редициа заемащ място от 2 колони и 1 редица.

Label userName = new Label("User Name:"); // създаване на обект от тип Label
grid.add(userName, 0, 1); // добавяне на Label-а в GridPane, на позиция 0
колона 1 редица

TextField userTextField = new TextField(); // създаване на обект от тип
```

```
TextField
grid.add(userTextField, 1, 1); //добавяне на TextField-а в GridPane, на
позиция 1 колона 1 редица

Label pw = new Label("Password:");
grid.add(pw, 0, 2);//добавяне на Label-а в GridPane, на позиция 0 колона 2
редица

PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);//добавяне на PasswordField-а в GridPane, на позиция 1
колона 2 редица
```

За сега приложението ни изглежда така:



Добавяне на бутон

```
Button btn = new Button("Sign in"); // създаване на обект от тип Button  с
текст в него Sign in
HBox hbBtn = new HBox(10); // създаване на Layout от тип HBox с разстояние
между елементите 10
hbBtn.setAlignment(Pos.BOTTOM_RIGHT); // задаване на позициониране в HBox-а
да е долу в дясно
hbBtn.getChildren().add(btn); // добавяне на бутона в HBox-а
grid.add(hbBtn, 1, 4); // добавяне на HBox-а в GridPane
```

Забележете, че за да добавим бутона първо взимам всички "деца" на HBox-а и добавяме бутона като негово "дете".

Добавяне празен текст, който ще служи за известяване, че сме влезли в системата

```
Text actiontarget = new Text(); // няма текст
 grid.add(actiontarget, 1, 6);
```

Добавяне на функционалност на бутона

```
btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent e) {
        System.out.println("Username: " + userTextField.getText());
        System.out.println("Password: " + pwBox.getText());
        actiontarget.setFill(Color.RED);
        actiontarget.setText("Sign in button pressed");
    }
});
```

В JavaFx за разлика от Swing се използва само един интерфейс, който наследявате за да се справите с събитията, които възникват в вашата програма - EventHandler. Единственото, което сменяте е вида на събитието, като го слагате като Generic тип на интерфейса.
Може да прочетете допълнително тук :
https://docs.oracle.com/javafx/2/events/convenience_methods.htm

А в конзолата ни излиза името и паролата.

Целият код: Файл:Login Form Fx.rar (импортирайте проекта в Eclipse File -> Open projects from file system)
Може да разгледате повече за Layout-ите и Control-и (Button, Text, Label, CheckBox и др.):
https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm
https://docs.oracle.com/javafx/2/ui_controls/overview.htm