Aula 16 - Desenvolvimento Desktop com JavaFX

Docupedia Export

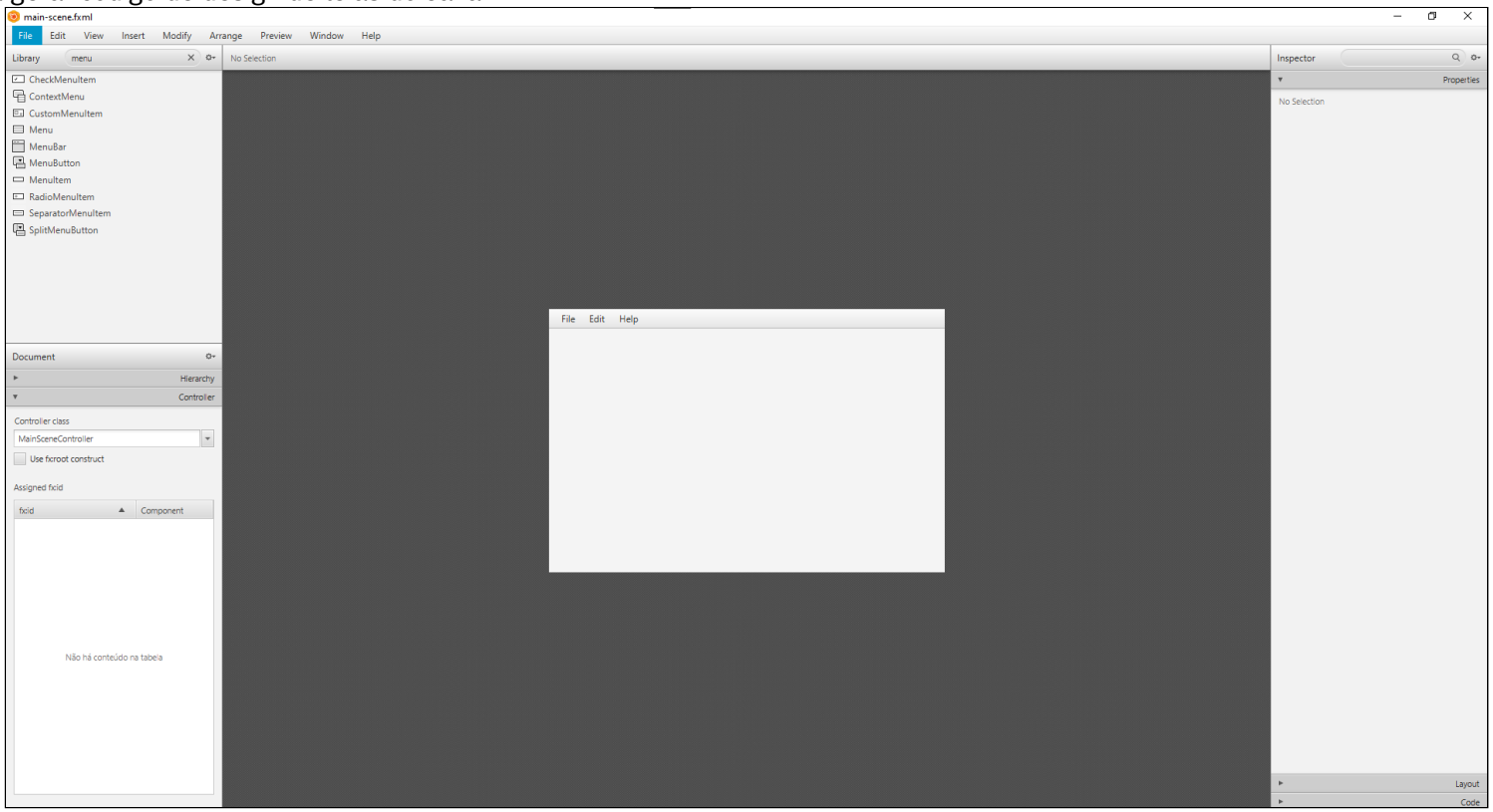
Author:Sobolevski Nycollas (CtP/ETS)
Date:07-Oct-2024 15:26

Table of Contents

- Obtendo o SceneBuilder
- Fazendo telas no SceneBuilder

1 Obtendo o SceneBuilder

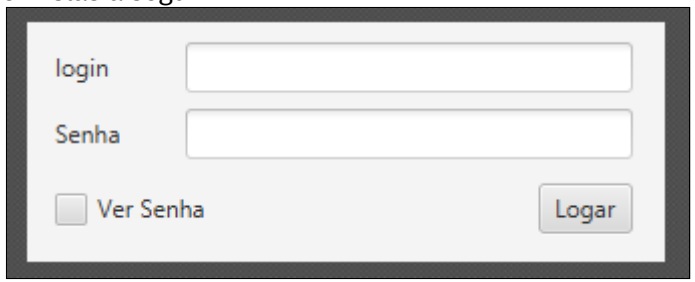
Vamos usar um software para nos ajudar a fazer nossas interfaces: https://gluonhq.com/products/scene-builder/thanks/?dl=/download/scene-builder-jar/. O Scene Builder nos permitirá gerar código de design de telas do JavaFX.

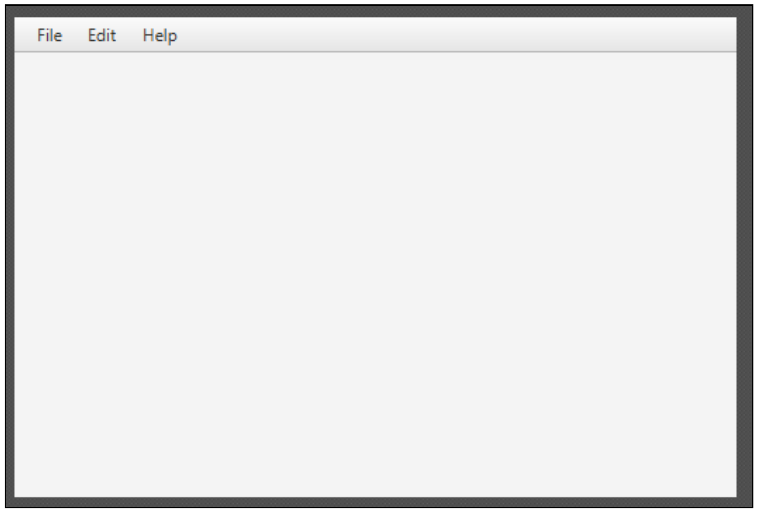


Com isso será muito fácil aprender a criar telas. Ao salvar o arquivo obteremos um arquivo do tipo fxml, um arquivo XML do java FX que descreve uma tela. A biblioteca do JavaFX nos dará formas de carregar esse arquivo no projeto. Observe a criação de um login simples, que antes parecia algo impossível e agora é consideravelmente fácil.

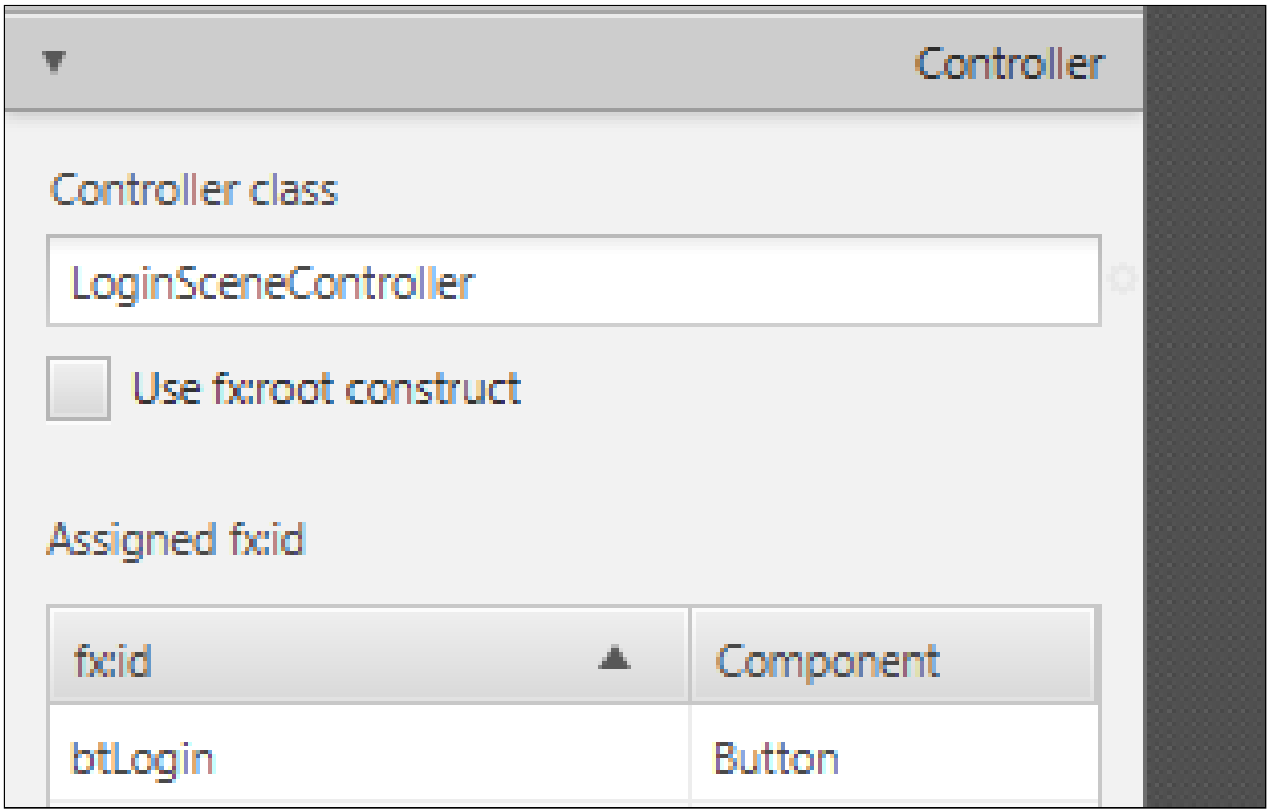
2 Fazendo telas no SceneBuilder

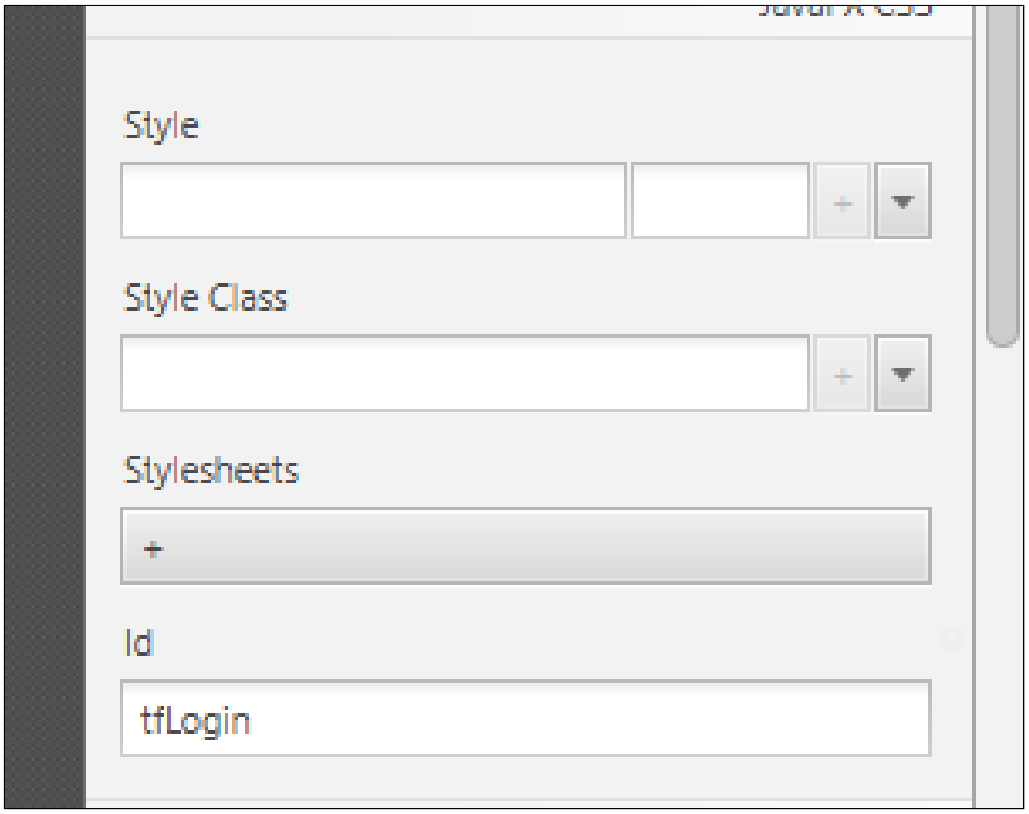
Vamos criar duas telas no SceneBuilder, elas podem ser vistas a seguir:

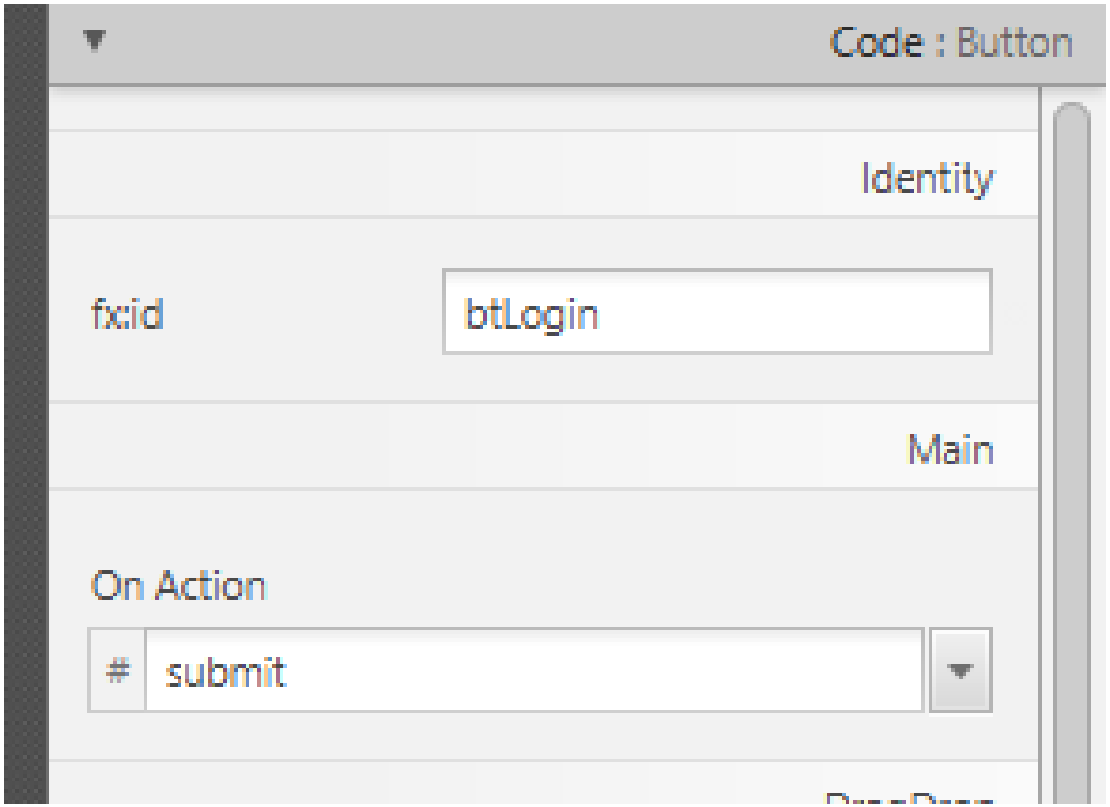




Além disso na aba **Controller** que fica a esquerda precisamos apontar qual será o controlador que interagirá com a tela no Java. Precisamos também indicar o nome de cada campo **id** na direita, na aba **Properties**, ao clicar em um componente. Isso dirá ao Java o nome da variável que usaremos na aplicação. Também podemos, na aba **Code** podemos colocar o nome das funções que gostaremos de usar:







Ao terminar de configurar e salvar na nossa pasta teremos o seguinte resultado:

login-scene.fxml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <?import javafx.scene.control.Button?>
4  <?import javafx.scene.control.CheckBox?>
5  <?import javafx.scene.control.Label?>
6  <?import javafx.scene.control.MenuBar?>
7  <?import javafx.scene.control.PasswordField?>
8  <?import javafx.scene.control.TextField?>
9  <?import javafx.scene.layout.AnchorPane?>
```

```
<?import javafx.scene.layout.VBox?>

<VBox prefHeight="119.0" prefWidth="318.0" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="LoginSceneController">
  <children>
    <MenuBar prefHeight="0.0" prefWidth="640.0" VBox.vgrow="NEVER" />
    <AnchorPane maxHeight="-1.0" maxWidth="-1.0" prefHeight="117.0" prefWidth="247.0" VBox.vgrow="ALWAYS">
        <children>
           <Button id="btLogin" fx:id="btLogin" layoutX="255.0" layoutY="80.0" mnemonicParsing="false" onAction="#submit" text="Logar" />
           <CheckBox id="cbPass" fx:id="cbPass" layoutX="14.0" layoutY="84.0" mnemonicParsing="false" text="Ver Senha" />
           <Label layoutX="14.0" layoutY="14.0" text="login" />
           <Label layoutX="14.0" layoutY="47.0" text="Senha" />
           <TextField id="tfLogin" fx:id="tfLogin" layoutX="79.0" layoutY="10.0" prefHeight="25.0" prefWidth="223.0" />
           <PasswordField id="pfPass" fx:id="pfPass" layoutX="79.0" layoutY="43.0" prefHeight="25.0" prefWidth="223.0" />
        </children>
    </AnchorPane>
  </children>
</VBox>
```

Agora vamos fazer a classe **LoginSceneController** que será efetivamente o controlador da aplicação:

LoginScreenController.java

```
import java.net.URL;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.stage.Stage;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.CheckBox;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
```

```
16
17  public class LoginSceneController {
18
19      // Vamos fazer uma função CreateScene que irá
20      // criar a cena apartir de um FXMLLoader carregando
21      // o .fxml.
22      public static Scene CreateScene() throws Exception
23      {
24          URL sceneUrl = LoginSceneController.class
25              .getResource("login-scene.fxml");
26          Parent root = FXMLLoader.load(sceneUrl);
27          Scene scene = new Scene(root);
28          return scene;
29      }
30
31      // Variáveis que representam os componentes
32      // Note que id/field devem ser iguais ao nome
33      // que aparece aqui.
34      @FXML
35      protected Button btLogin;
36
37      @FXML
38      protected TextField tfLogin;
39
40      @FXML
41      protected PasswordField pfPass;
42
43      @FXML
44      protected CheckBox cbPass;
45
46      // Evento submit executado ao rodar a aplicação.
47      @FXML
48      protected void submit(ActionEvent e) throws Exception {
49          if (!tfLogin.getText().equals("don")) {
50              Alert alert = new Alert(
51                  AlertType.ERROR,
52                  "Você não é o don!",
53                  ButtonType.OK
54              );
```

```
            alert.showAndWait();
            return;
        }

        if (!pfPass.getText().equals("verstapi")) {
            Alert alert = new Alert(
                AlertType.ERROR,
                "Você não é o don!",
                ButtonType.OK
            );
            alert.showAndWait();
            return;
        }

        // Fechando o login
        Stage crrStage = (Stage)btLogin
            .getScene().getWindow();
        crrStage.close();

        // Abrindo a tela principal
        Stage stage = new Stage();
        Scene scene = MainSceneController.CreateScene();
        stage.setScene(scene);
        stage.show();
    }
}
```

Vamos fazer isso também para tela principal que no caso está bem vazia:

MainScreenController.java

```
import java.net.URL;
import javafx.scene.Scene;
import javafx.scene.Parent;
import javafx.fxml.FXMLLoader;

public class MainSceneController {

    public static Scene CreateScene() throws Exception
```

```
    {
        URL sceneUrl = LoginSceneController.class
            .getResource("main-scene.fxml");
        Parent root = FXMLLoader.load(sceneUrl);
        Scene scene = new Scene(root);
        return scene;
    }
}
```

E é claro, vamos fazer uma classe **App** que gerenciará e abrirá o programa junto da nossa função **Main**:

App.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        Scene scene = LoginSceneController.CreateScene();
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

E assim criamos de forma dinâmica e simples uma aplicação realmente simples. Para executar vamos fazer um script em PowerShell. O problema é que por vezes algum controlador pode não ser compilado a depender de como você o utiliza (se você não fizer um CreateScene() e usar o FXMLLoader diretamente). Outro problema é que os .fxml devem estar na pasta junto com os .class. Assim observe um script simples feito para execução do projeto:

run.ps1

```
clear
```

```
2
3  # Se a bin não existir, a cria
4  $binCount = ls | ? { $_.Name -eq "bin" } | measure | % { $_.Count }
5  if ($binCount -eq "0")
6  {
7      "Criando pasta bin..."
8      mkdir bin
9  }
10
11 # Para cada arquivo na pasta que terminal com .fxml, copia para bin
12 ls | ? { $_.Name.EndsWith(".fxml") } | % { cp $_.Name .\bin\ }
13
14 # Para cada arquivo na pasta que contém Controller no nome, copia para bin
15 ls | ? { $_.Name.Contains("Controller") } | % { javac -d bin $_.Name}
16
17 # Compila e executa o App.java
18 javac App.java -d bin
19 cd bin
20 java App
21 cd ..
```