JavaFX Studies/ Aula07: Entendendo sobre Logout e a Classe Alert

Primeiramente seguimos o padrão de criação de um programa, fazemos toda a estilização, organização de pastas e etc, porém o que é importante para a ação de saida do programa é que o nosso AnchorPane tenha um ID, e nosso botão de logout tenha um id também e um método setado no seu OnAction, afinal quando presssionado ele vai realizar uma ação, segue abaixo o código do controlador de uma tela que possui apenas um pane, e um botão de logout.

```
public class Controller07 {
    @FXML
    private Button logoutButton;

    @FXML
    private AnchorPane pane;

    Stage stage;

    public void logout(ActionEvent event){

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle(title:"Logout");
        alert.setHeaderText(headerText:"Você deseja mesmo
        sair?");
        alert.setContentText(contentText:"Clique em OK para
        confirmar ou em Cancelar para voltar.");

        if(alert.showAndWait().get() == ButtonType.OK){
            stage = (Stage) pane.getScene().getWindow();
            stage.close();
        }
    }


}
```

Nessa tela fiz a injeção do Button e do AnchorPane, e junto a isso declarei uma variável do tipo Stage.

Quando chamo o método logout ele é dividido em 2 partes: A parte que controla a saida com o Alert e a segunda parte que seria o real código que realiza o fechamento da janela, do Stage no caso.

Como funciona o Alert: primeiramente temos que criar o Alert e declara seu tipo, nesse caso foi criado com o tipo Alert.Alerttype.CONFIRMATION, logo é uma janela que possui um botão para confirmar ou cancelar, aqui estão alguns tipos de Alert:

- **Information:**
  - **Tipo:** Alert.AlertType.INFORMATION
  - **Descrição:** Uma caixa de diálogo informativa que exibe uma mensagem simples.
- **Confirmation:**
  - **Tipos:** Alert.AlertType.CONFIRMATION
  - **Descrição:** Uma caixa de diálogo de confirmação usada para obter a confirmação do usuário.
- **Warning:**
  - **Tipo:** Alert.AlertType.WARNING
  - **Descrição:** Uma caixa de diálogo de aviso para alertar o usuário sobre uma situação potencialmente problemática.
- **Error:**
  - **Tipo:** Alert.AlertType.ERROR
  - **Descrição:** Uma caixa de diálogo de erro para exibir mensagens de erro ao usuário.
- **None:**
  - **Tipo:** Alert.AlertType.NONE
  - **Descrição:** Um alerta personalizado que não exibe um ícone padrão.
- **TextInputDialog:**
  - **Descrição:** Uma caixa de diálogo que permite ao usuário inserir um texto.
- **ChoiceDialog:**
  - **Descrição:** Uma caixa de diálogo que permite ao usuário escolher uma opção a partir de uma lista de opções.

Exemplo de uso básico de um Alert:

```
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;

public class Main {
    public static void main(String[] args) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Informação");
        alert.setHeaderText("Este é um cabeçalho opcional");
        alert.setContentText("Esta é a mensagem principal do alerta.");

        // Exibindo o alerta e aguardando a resposta do usuário
        alert.showAndWait();
    }
}
```

Logo após a utilização do Alert eu checo a escolha do usuário, nesse caso com o if, o que esse if faz é mostrar o alerta e esperar uma resposta do usuário(.showAndWait), e quando essa resposta for obtida ele pega essa resposta(.get), e faz a comparação decretada pelo usuário, nesse caso ele testa se o usuário apertou o OK button, caso ele tenha pressionado, a minha variável Stage declarada no início recebe com um castin o a janela que o nosso pane está e logo após realiza o Stage.close(), para fechar a janela.

Agora vamos usar esse mesmo metódo de logout porém quando o usuário clicar no X para fechar ou dar Atl+F4:

Primeiramente o mesmo método é usado porém na classe main do programa, mas algumas alterações devem ser feitas, como o parâmetro que esse método vai receber, esse método vai receber agora só um Stage, pq n terá um ActionEvent para dispara-ló, assim o método ficaria da seguinte forma:

```
public void logout(Stage stage){

    Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
    alert.setTitle(title:"Logout");
    alert.setHeaderText(headerText:"Você deseja mesmo
    sair?");
    alert.setContentText(contentText:"Clique em OK para
    confirmar ou em Cancelar para voltar.");

    if(alert.showAndWait().get() == ButtonType.OK){
        stage.close();
    }
}
```

Ele recebe um Stage, que é o Stage que está sofrendo a ação de tentativa de fechamento que será passado como o atual, vamos demonstrar agora no código:

```java
@Override
public void start(Stage primaryStage) throws Exception{

    Parent root = FXMLLoader.load(getClass().getResource
    (name:"./view/tela_aula07.fxml"));

    Image icon = new Image(getClass().getResourceAsStream
    (name:"./assets/icon1.png"));

    Scene scene = new Scene(root);

    String css = this.getClass().getResource(name:"./css/
    aula07.css").toExternalForm();
    scene.getStylesheets().add(css);

    primaryStage.setOnCloseRequest(event -> {
        event.consume();
        logout(primaryStage);
    });

    primaryStage.setScene(scene);
    primaryStage.getIcons().add(icon);
    primaryStage.setTitle(value:"Aula 07 - JavaFX");
    primaryStage.show();
}
```

Logo no meu método Start eu irei no primaryStage, chamar o método .setOnCloseRequest, que modifica a ação que será realizada quando o usuário tenta fechar a janela.

Nesse método eu vou criar uma lambda function, que vai chamar o método logout, passando como parâmetro o primaryStage, que é o Stage atual que o usuário tentou fechar, mas junto a isso eu necessito antes chamar o método event.consume();, para entender como o event.consume(); funciona precisamos entender sobre e eventos e ouvintes no JavaFx, aqui tentarei explicar brevemente como funciona a lógica dos eventos e dos ouvintes nesse caso de tentativa de fechamento de um Stage:

O método setOnCloseRequest é usado para definir um ouvinte de eventos que será acionado quando a janela principal (primaryStage) estiver prestes a ser fechada, o evento OnCloseRequest é gerado quando o usuário tenta fechar a janela, ao chamar event.consume(), você está indicando que o evento de fechamento deve ser consumido, impedindo que outros ouvintes de eventos associados a esse evento sejam notificados, isso é feito antes de chamar o método logout(primaryStage), o objetivo é garantir que a lógica de logout seja executada antes que a janela seja realmente fechada, se event.consume() não fosse chamado, a janela continuaria seu processo de fechamento padrão após a execução do método logout(primaryStage), logo consumir o evento aqui permite que você tenha controle sobre o momento exato em que a janela é fechada, executando a lógica de logout primeiro.

#ID