Curso: Java COMPLETO - Programação Orientada a Objetos + Projetos

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Capítulo: Projeto JavaFX com JDBC

Objetivo geral:

- Introduzir o aluno ao desenvolvimento de aplicações JavaFX com JDBC
- Permitir que o aluno conheça os fundamentos e a utilização das ferramentas, de modo que ele possa depois prosseguir estudando, de forma confortável, as especificidades que desejar

  REQUISITOS: OO & Lambda & JDBC & JavaFX

  PROJETO: https://github.com/acenelio/workshop-javafx-jdbc

Github project

Checklist:

- Gitignore: Java

Local project created

Checklist:

- User libraries: JavaFX, MySQLConnector
- Run configurarions -> VM arguments:
  --module-path C:\java-libs\javafx-sdk\lib --add-modules=javafx.fxml,javafx.controls
- Git:
  - git init
  - git remote add origin https://github.com/acenelio/workshop-javafx-jdbc.git
  - git pull origin master
- Gitignore:

```
# build folders
bin/
target/
nbproject/private/
build/
nbbuild/
dist/
nbdist/
```

Main view

Checklist:

- Create FXML "MainView" (package "gui")
- Load FXML in Main
- Update Main.java

```java
@Override
public void start(Stage primaryStage) {
	try {
		FXMLLoader loader = new FXMLLoader(getClass().getResource("/gui/MainView.fxml"));
		Parent parent = loader.load();

		Scene mainScene = new Scene(parent);
		primaryStage.setScene(mainScene);
		primaryStage.setTitle("Sample JavaFX application");
		primaryStage.show();
	} catch (IOException e) {
		e.printStackTrace();
	}
}
```

Main view design

Checklist:

- Design MainView.fxml
- Customize menu items
- Update Main.java

```xml
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.layout.VBox?>


<ScrollPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/10.0.1"
xmlns:fx="http://javafx.com/fxml/1">
   <content>
      <VBox prefHeight="326.0" prefWidth="513.0">
         <children>
            <MenuBar>
              <menus>
                <Menu mnemonicParsing="false" text="Registration">
                  <items>
                    <MenuItem mnemonicParsing="false" text="Seller" />
                        <MenuItem mnemonicParsing="false" text="Departments" />
                  </items>
                </Menu>
                <Menu mnemonicParsing="false" text="Help">
                  <items>
                    <MenuItem mnemonicParsing="false" text="About" />
                  </items>
                </Menu>
              </menus>
            </MenuBar>
         </children>
      </VBox>
   </content>
</ScrollPane>
```

Main view controller

Checklist:

- Create controller
- In view, associate controller, ids, events

About view

Checklist:

- Include util classes to the project (Alerts.java, Constraints.java)
  https://github.com/acenelio/javafx5/blob/master/src/gui/util/Alerts.java
  https://github.com/acenelio/javafx5/blob/master/src/gui/util/Constraints.java
- Create About.fxml (VBox)
- In Main.java, expose mainScene reference
- In MainViewController.java, create loadView method

DepartmentList view design

Checklist:

- Create DepartmentList.fxml (VBox)
- In MainViewController.java, load DepartmentList

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.ToolBar?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/10.0.1"
xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <Label text="Department Registration">
         <font>
            <Font name="System Bold" size="14.0" />
         </font>
         <padding>
            <Insets left="5.0" top="5.0" />
         </padding>
      </Label>
      <ToolBar prefHeight="40.0" prefWidth="200.0">
        <items>
          <Button fx:id="btNew" mnemonicParsing="false" text="New" />
        </items>
      </ToolBar>
      <TableView prefHeight="200.0" prefWidth="200.0">
        <columns>
          <TableColumn prefWidth="75.0" text="Id" />
          <TableColumn prefWidth="75.0" text="Name" />
        </columns>
      </TableView>
   </children>
</VBox>
```

DepartmentList controller

Checklist:

- Create model.entities.Department.java
  https://github.com/acenelio/demo-dao-jdbc/blob/master/src/model/entities/Department.java
- Create DepartmentListController.java
- In view, associate controller, ids, events

DepartmentService

Checklist:

- Create model.services.DepartmentService.java with findAll method
- In DepartmentListController:
  - Create DepartmentService dependency with set method
  - Create ObservableList<Department>
  - Create updateTableViewData method

Initializing action as parameter

Checklist:

- Add a Consumer<T> parameter to loadView method
- After loading view, call accept from the Consumer
- Add a consumer instance on loadView calls

Adding database access

Prerequisites:

- MySQL server installed and running
- Database created and instantiated
  https://github.com/acenelio/demo-dao-jdbc/blob/master/database.sql
- Data access layer implemented (DAO pattern):
  https://github.com/acenelio/demo-dao-jdbc

Checklist:

- Add model.entities.Seller.java
- Add db.properties do project
- Add data access packages to project:
  - db
  - model.dao
  - model.dao.impl
- In DepartmentService, add DepartmentDao dependency with Factory call

DepartmentForm (dialog) design

Checklist:

- Create gui.util.Utils.java with currentStage method
- Create DepartmentForm.fxml (AnchorPane)
  - GridPane 3x3 (anchors: 20 top, 20 left)
  - Id text box: not editable
  - Label error: red
  - HBox (spacing: 5)
- In DepartmentListController, create createDialogForm method
- Call createDialogForm on "new" button action

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>

<AnchorPane prefHeight="139.0" prefWidth="532.0" xmlns="http://javafx.com/javafx/10.0.1"
xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <GridPane layoutX="33.0" layoutY="32.0" prefHeight="95.0" prefWidth="499.0"
AnchorPane.leftAnchor="20.0" AnchorPane.topAnchor="20.0">
        <columnConstraints>
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="161.0" minWidth="10.0" prefWidth="63.0" />
          <ColumnConstraints hgrow="SOMETIMES" maxWidth="270.0" minWidth="10.0" prefWidth="160.0" />
            <ColumnConstraints hgrow="SOMETIMES" maxWidth="245.0" minWidth="10.0" prefWidth="243.0" />
        </columnConstraints>
        <rowConstraints>
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
          <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        </rowConstraints>
         <children>
            <Label text="Id" />
            <Label text="Name" GridPane.rowIndex="1" />
            <Label textFill="RED" GridPane.columnIndex="2" GridPane.rowIndex="1" />
            <TextField editable="false" GridPane.columnIndex="1" />
            <TextField GridPane.columnIndex="1" GridPane.rowIndex="1" />
            <HBox prefHeight="100.0" prefWidth="200.0" spacing="5.0" GridPane.columnIndex="1"
GridPane.rowIndex="2">
               <children>
                  <Button mnemonicParsing="false" text="Save" />
                  <Button mnemonicParsing="false" text="Cancel" />
               </children>
            </HBox>
         </children>
      </GridPane>
   </children>
</AnchorPane>
```

DepartmentFormController

Checklist:

- Create DepartmentFormController.java
- In view, associate controller, ids, events

Passing a Department object to DepartmentForm view

Checklist:

- In DepartmentFormController
  - Create a Department dependency with set method
  - Create updateFormData method
- In DepartmentListController
  - Update onBtNewAction method
  - Update createDialogForm method

Saving a new Department

Checklist:

- In Utils, implement tryParseToInt method
- In DepartmentService, create saveOrUpdate method
- In DepartmentFormController
  - Create a DepartmentService dependency with set method
  - Implement onBtSaveAction
  - Implement onBtCancelAction
- In DepartmentListController, inject DepartmentService instance

Observer design pattern to update tableview

Checklist:

- Create interface gui.listeners.DataChangeListener
- In DepartmentFormController (subject)
  - Create List<DataChangeListener> dependency with subscribe method
  - Notify subscribers when needed
- In DepartmentListController (observer)
  - Implement DataChangeListener interface
  - Subscribe for DepartmentFormController

Validation exception

Checklist:

- Create model.exceptions.ValidationException
- In DepartmentFormController
  - In getFormData method, implement verifications and throw ValidationException
  - Implement setErrorMessages method
  - In onBtSaveAction, catch ValidationException

Update department

References:

https://stackoverflow.com/questions/32282230/fxml-javafx-8-tableview-make-a-delete-button-in-each-row-and-delete-the-row-a

Checklist:

- In DepartmentListController
  - Create new attribute: TableColumn<Department, Department> tableColumnEDIT;
  - Create initEditButtons method
  - In updateTableViewData, call initEditButtons
- In DepartmentList.fxml
  - Include new table column
  - Associate id

```
private void initEditButtons() {
	tableColumnEDIT.setCellValueFactory(param -> new ReadOnlyObjectWrapper<>(param.getValue()));
	tableColumnEDIT.setCellFactory(param -> new TableCell<Department, Department>() {
		private final Button button = new Button("edit");

		@Override
		protected void updateItem(Department obj, boolean empty) {
			super.updateItem(obj, empty);

			if (obj == null) {
				setGraphic(null);
				return;
			}

			setGraphic(button);
			button.setOnAction(
			event -> createDialogForm(
				obj, "/gui/DepartmentForm.fxml",Utils.currentStage(event)));
		}
	});
}
```

Remove department

References:

https://stackoverflow.com/questions/32282230/fxml-javafx-8-tableview-make-a-delete-button-in-each-row-and-delete-the-row-a

Checklist:

- In Alerts, create showConfirmation method
- In DepartmentService, create remove method
- In DepartmentListController
  - Create new attribute: TableColumn<Department, Department> tableColumnREMOVE;
  - Create initRemoveButtons method
    - Catch DbIntegrityException
  - In updateTableViewData, call initRemoveButtons
- In DepartmentList.fxml
  - Include new table column
  - Associate id

```
public static Optional<ButtonType> showConfirmation(String title, String content) {
        Alert alert = new Alert(AlertType.CONFIRMATION);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(content);
        return alert.showAndWait();
}



private void initRemoveButtons() {
        tableColumnREMOVE.setCellValueFactory(param -> new ReadOnlyObjectWrapper<>(param.getValue()));
        tableColumnREMOVE.setCellFactory(param -> new TableCell<Department, Department>() {
         private final Button button = new Button("remove");

         @Override
         protected void updateItem(Department obj, boolean empty) {
             super.updateItem(obj, empty);

             if (obj == null) {
                 setGraphic(null);
                 return;
             }

             setGraphic(button);
             button.setOnAction(event -> removeEntity(obj));
         }
    });
}
```