Login Form with JavaFX

BY: PROFESSOR WERGELES

Creating a form in JavaFX

- Creating a form is a common activity when developing an application
- This tutorial teaches you the basics of
  - screen layout
  - how to add controls to a layout pane
  - and how to create input events
- In this tutorial, you will use JavaFX to build the login form shown in Figure 2-1

Figure 2-1 Login Form

Create a GridPane Layout

- For the login form, use a GridPane layout because it enables you to create a flexible grid of rows and columns in which to lay out controls
- You can place controls in any cell in the grid, and you can make controls span cells as needed
- The code to create the GridPane layout is in Example 2-2

Example 2-2 GridPane with Gap and Padding Properties

```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));

Scene scene = new Scene(grid, 300, 275);
primaryStage.setScene(scene);
```

Create a GridPane Layout

- Example 2-2 creates a GridPane object and assigns it to the variable named grid
  - The alignment property changes the default position of the grid from the top left of the scene to the center
  - The gap properties manage the spacing between the rows and columns, while the padding property manages the space around the edges of the grid pane
  - The insets are in the order of top, right, bottom, and left
    - In this example, there are 25 pixels of padding on each side.
- The scene is created with the grid pane as the root node, which is a common practice when working with layout containers
  - Thus, as the window is resized, the nodes within the grid pane are resized according to their layout constraints
  - In this example, the grid pane remains in the center when you grow or shrink the window
  - The padding properties ensure there is a padding around the grid pane when you make the window smaller.
- This code sets the scene width and height to 300 by 275
  - If you do not set the scene dimensions, the scene defaults to the minimum size needed to display its contents.

Coding Exercise

- Now add the GridPane to our code

Add Text, Labels, and Text Fields

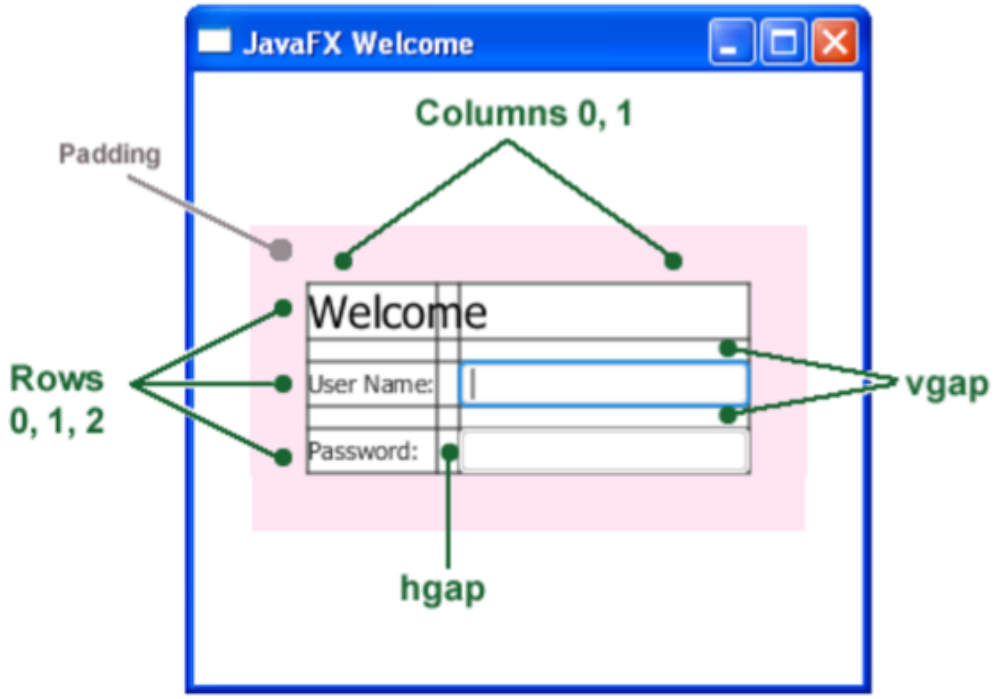
Example 2-3 Controls

```
Text scenetitle = new Text("Welcome");
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
grid.add(scenetitle, 0, 0, 2, 1);

Label userName = new Label("User Name:");
grid.add(userName, 0, 1);

TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);

Label pw = new Label("Password:");
grid.add(pw, 0, 2);

PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);
```

- Note: When working with a grid pane, you can display the grid lines, which is useful for debugging purposes: grid.setGridLinesVisible(true)

Login Form with Grid Lines

Figure 2-2 Login Form with Grid Lines



Coding Exercise

- Now add these elements to our code

Add a Button and Text

- First, create the button and position it on the bottom right, which is a common placement for buttons that perform an action affecting the entire form

Example 2-4 Button

```
Button btn = new Button("Sign in");
HBox hbBtn = new HBox(10);
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 4);
```

- The HBox pane sets an alignment for the button that is different from the alignment applied to the other controls in the grid pane
  - The alignment property has a value of Pos.BOTTOM_RIGHT, which positions a node at the bottom of the space vertically and at the right edge of the space horizontally
  - The button is added as a child of the HBox pane, and the HBox pane is added to the grid in column 1, row 4.

Add a Button and Text

- Now, add a Text control for displaying the message

Example 2-5 Text

```
final Text actiontarget = new Text();
        grid.add(actiontarget, 1, 6);
```

Figure 2-3 Login Form with Button

Add Code to Handle an Event

- Finally, make the button display the text message when the user presses it

Example 2-6 Button Event

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

    @Override
    public void handle(ActionEvent e) {
        actiontarget.setFill(Color.FIREBRICK);
        actiontarget.setText("Sign in button pressed");
    }
});
```

- The setOnAction() method is used to register an event handler that sets the actiontarget object to Sign in button pressed when the user presses the button
  - The color of the actiontargetobject is set to firebrick red

Run the Application

Figure 2-4 Final Login Form

Coding Exercise

- Finish up our code, add the remaining elements, and run it