Clearly it is going to involved a lot of work

To implement anything more than a trivial interface

Luckily there are tools available to support this task

Scene Builder

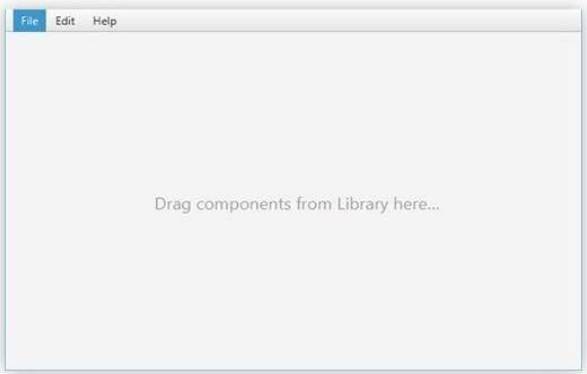
There is a tool called "Scene Builder" from Gluon
An application for building JavaFX GUI scenes:

https://gluonhq.com/products/scene-builder/

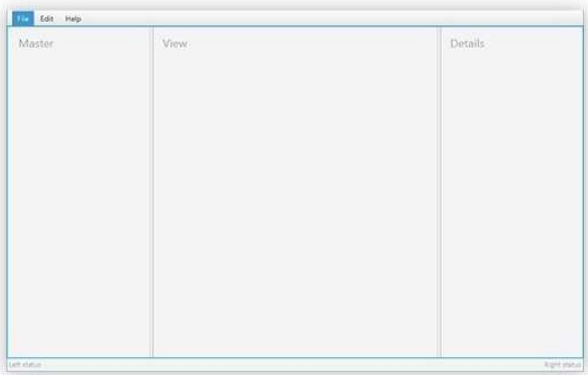
Let's do a live demo of the tool
(Slides provided for future reference)

Starting a new Scene

Desktop



Basic Application



Complex Application

Mobile

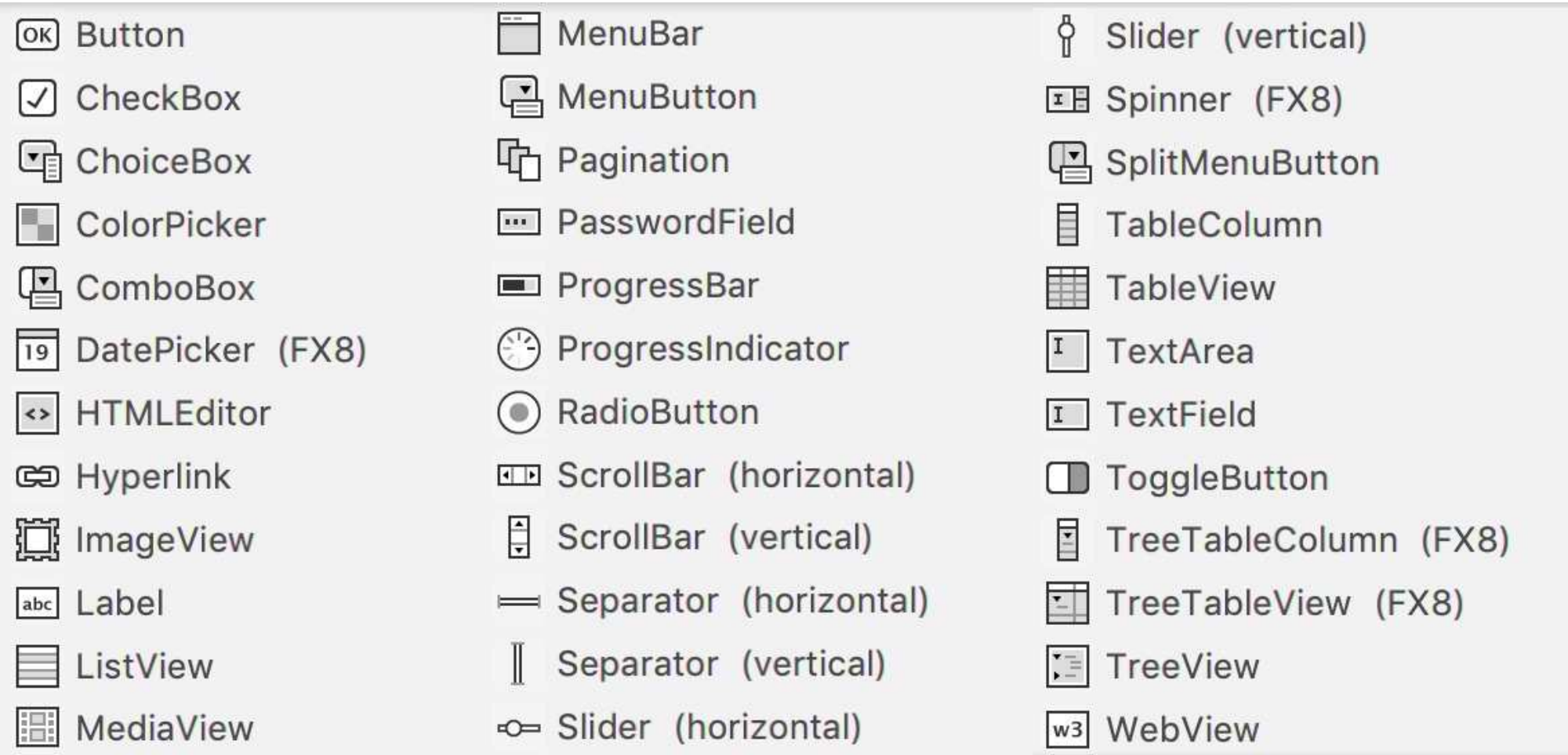
Empty Screen

Basic Screen

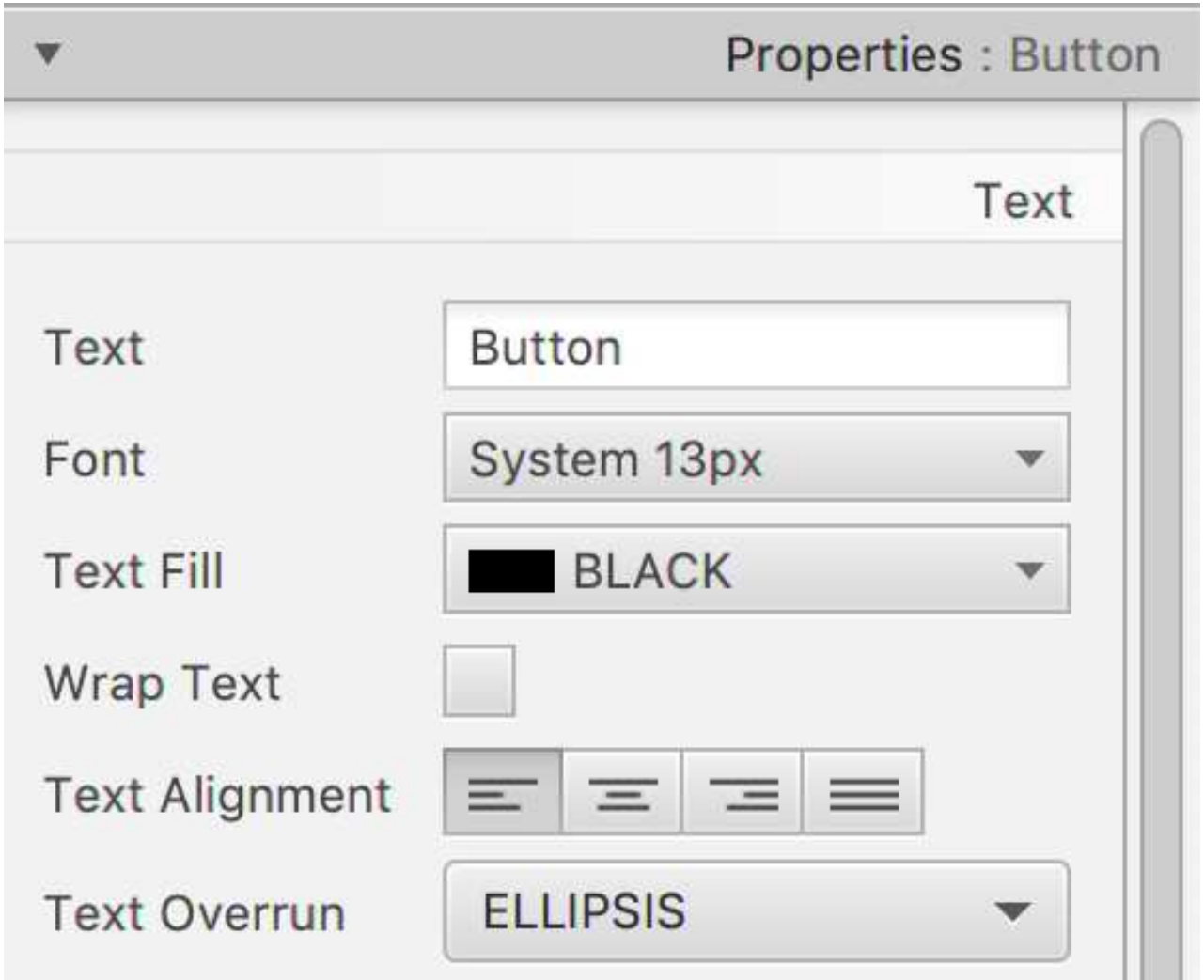
The Container Palette

- Accordion
- Accordion (empty)
- AnchorPane
- BorderPane
- ButtonBar (FX8)
- DialogPane (empty) (FX8)
- DialogPane (FX8)
- FlowPane
- GridPane
- HBox
- Pane
- ScrollPane
- ScrollPane (empty)
- SplitPane (empty)
- SplitPane (horizontal)
- SplitPane (vertical)
- StackPane
- Tab
- TabPane
- TabPane (empty)
- TextFlow (FX8)
- TilePane
- TitledPane
- TitledPane (empty)
- ToolBar
- VBox

The Control Palette



Control Properties

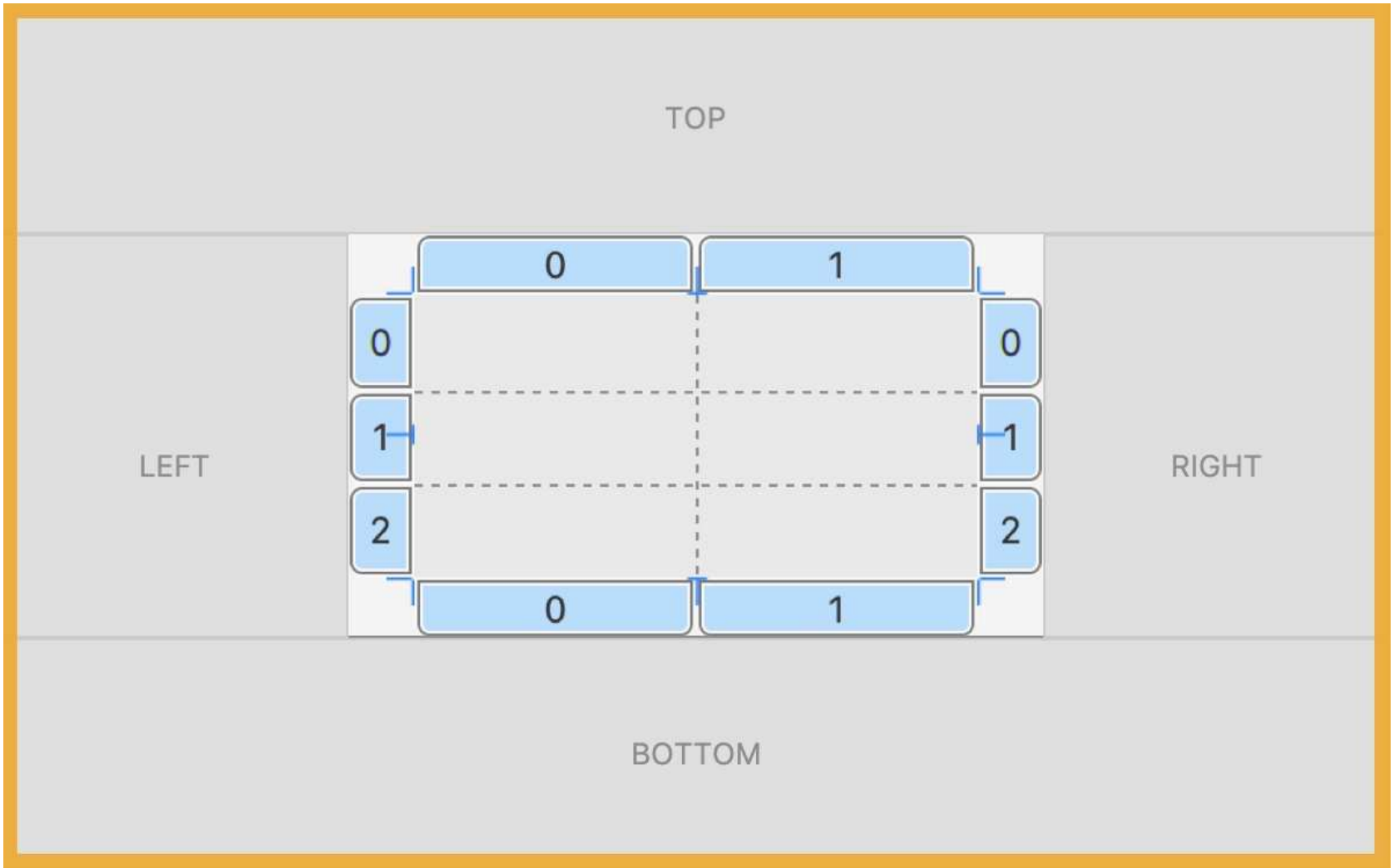


Different Widths and Heights

- Minimum: How small can we squash when scaling
- Preferred: What size should widget be normally
- Maximum: How big widget can get when scaling

- Use preferred size: Preferred size as max & min
- Use computed size: Let layout pane decide !
  (If a pane, pack to the preferred size of content)
- Enter number: Hardcode a particular size

Nesting Panes



Space

Within "Scene Builder" (and JavaFX in general),
There are various different notions of "space":

- Padding: Make widget itself fatter or slimmer
- Margin: Create blank space around widget
- Spacing: Gap between elements (HBox & VBox)

Previewing Your Interface

There is a preview feature in Scene Builder

Which can be used to view a running interface

Allows you to check various dynamic features

(scrollers, sliders, popups, menus etc)

Also check that everything resizes correctly

Saving your Interface

Scene Builder doesn't save interfaces as Java !
Rather, it exports them as a form of XML

It separates front-end "view" from behaviour logic
Just like with HTML and Javascript

Makes it easier to import GUIs into other tools
Alter behaviour without impacting appearance
Or change appearance without impacting behaviour
(Essential if we are to use Scene Builder later)

FXML

Markup language for specifying JavaFX GUIs

```
<BorderPane>
    <left>
        <Label text="Find what" />
    </left>
    <center>
        <TextField>
            <BorderPane.margin>
                <Insets left="10.0" right="10.0" />
            </BorderPane.margin>
        </TextField>
    </center>
    <right>
        <Button prefWidth="65.0" text="Find" />
    </right>
</BorderPane>
```

Importing FXML into Java

```
InputStream stream = new FileInputStream(filename);
FXMLLoader loader = new FXMLLoader();
Parent content = loader.load(stream);
Scene scene = new Scene(content);
stage.setScene(scene);
stage.show();
```

FXMLLoaderJFX

Event Handling

Defining up Event Handlers

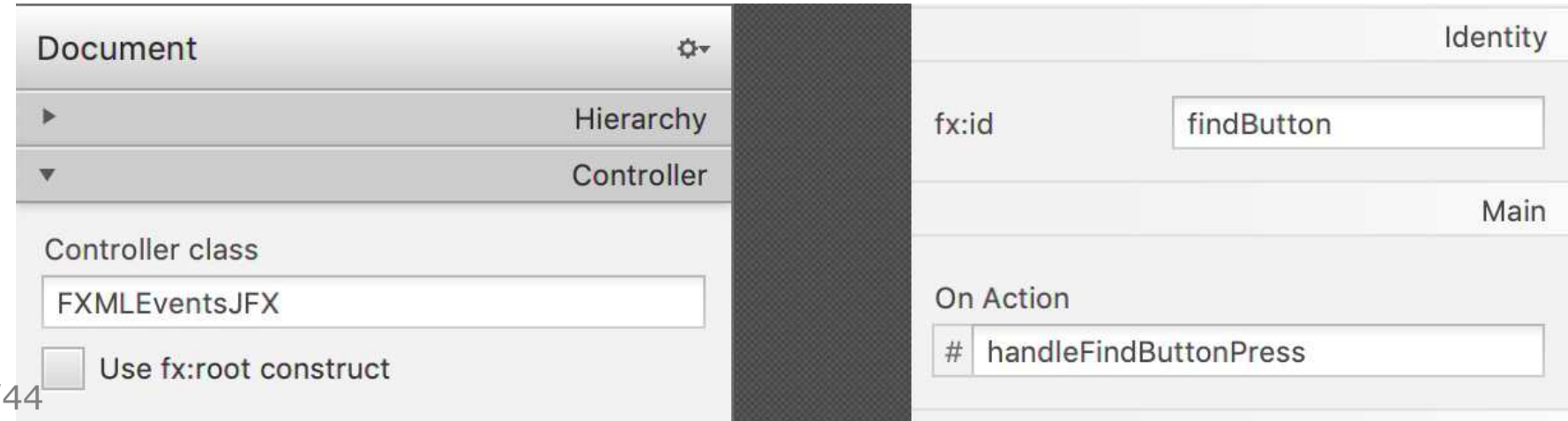
Define a Controller Class for the whole interface

(In the "Document" section on the Left-Hand-Side)

For the Widget, give it a unique ID and...

Enter the name of the Handler method

(In the "Code" section on the Right-Hand-Side)



Hooking up Event Handlers

All we have to do then is to
Write the previously named handler method
Inside the previously named Controller class:

```
public void handleFindButtonPress(ActionEvent event)
{
   println("Find button pressed !!!");
}
```

FXMLEventsJFX

Advantages of Scene Builder

- All of the controls are listed on a palette
- No need to look up attribs. and methods

  (They are all shown for you to pick from)
- Each type of pane is presented differently

  (So it is clear how it operates)
- Attaching event handlers is relative simple

Disadvantages of Scene Builder

- Scene Builder tool can be fiddly to use
- All options are presented - can be overwhelming
- Tool provides an additional learning overhead
- Extra layer of complexity (FXML)
- You still need to learn about JavaFX controls (e.g. how all the layout panes work)

It's not perfect, but...

It's the best GUI editor I have seen so far !