BAHMANM.COM
CARVING DEEP INTO IT, INFORMATION SYSTEMS AND ERP

Spec - Part II: The Layout

GUI Development in Pharo Smalltalk using Spec. It covers the layouts and tidies up the Greeter application built in part I.

Contents

Introduction

First of all: have you read the first part of this series yet? If not, please do so as we are going to iterate over what was done there.

As you recall, the greeter application which you built earlier was fully functional. However, it was not easy on the eye --in other words, it was very ugly. As promised, we will tidy it up in this episode.

The Question Of The Ages

"What do you want!?" Before setting off on the road of endlessly pushing and pulling widgets around, let's first settle for a target design. For the rest of this episode, we will try to achieve the following design (this design is done in LibreOffice Draw and is not a snapshot from the system):

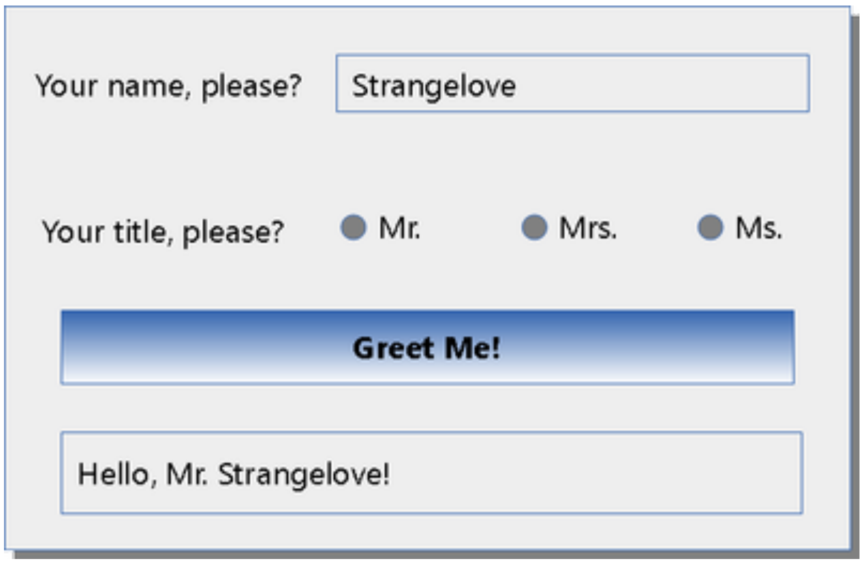

Figure 1 – The target design for this episode

Layout Smithing Tools

There are about half a dozen of layouts and widgets (which can be used to layout other widgets) in Spec. But for the sake of simplicity, only the two most basic ones are used in this episode:

1. `SpecColumnLayout`: You're already familiar with this layout as it was the only one used in part I. It is, as the name implies, a column. Widgets will be added vertically, from top to bottom; meaning the first widget which is added using `add:` will be at the top, and the last widget at the bottom of the column.
2. `SpecRowLayout`: Very similar to SpecColumnLayout but everything is done horizontally. Widgets will be added horizontally, from left to right; meaning the first widget which is added using `add:` will be at the left, and the last widget at the right of the row.

Talk To Me In Rows & Columns!

It's time to breakdown the target design to see how it can be achieved using our two layout tools. As you have already spotted, there are 4 rows in the design.

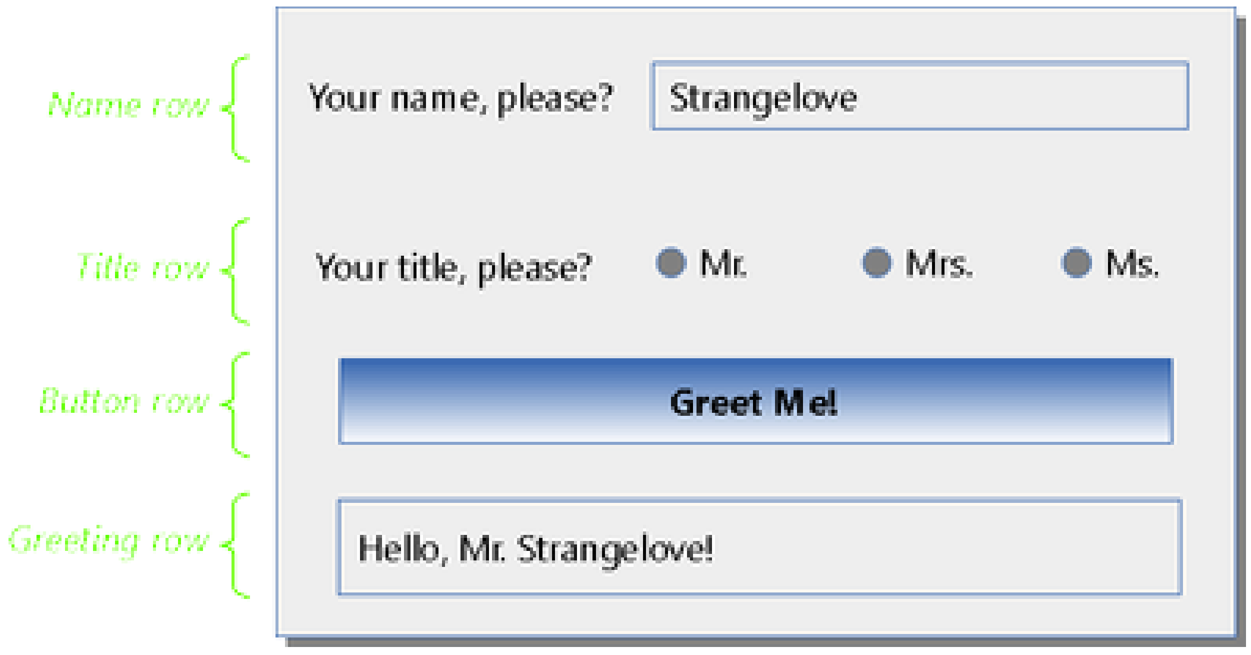

Figure 2 – Breakdown by rows

So, obviously, we can use 4 `SpecRowLayout`s to achieve this. Also, you can see that the titles are themselves neatly arranged in their own row inside *Title row*.

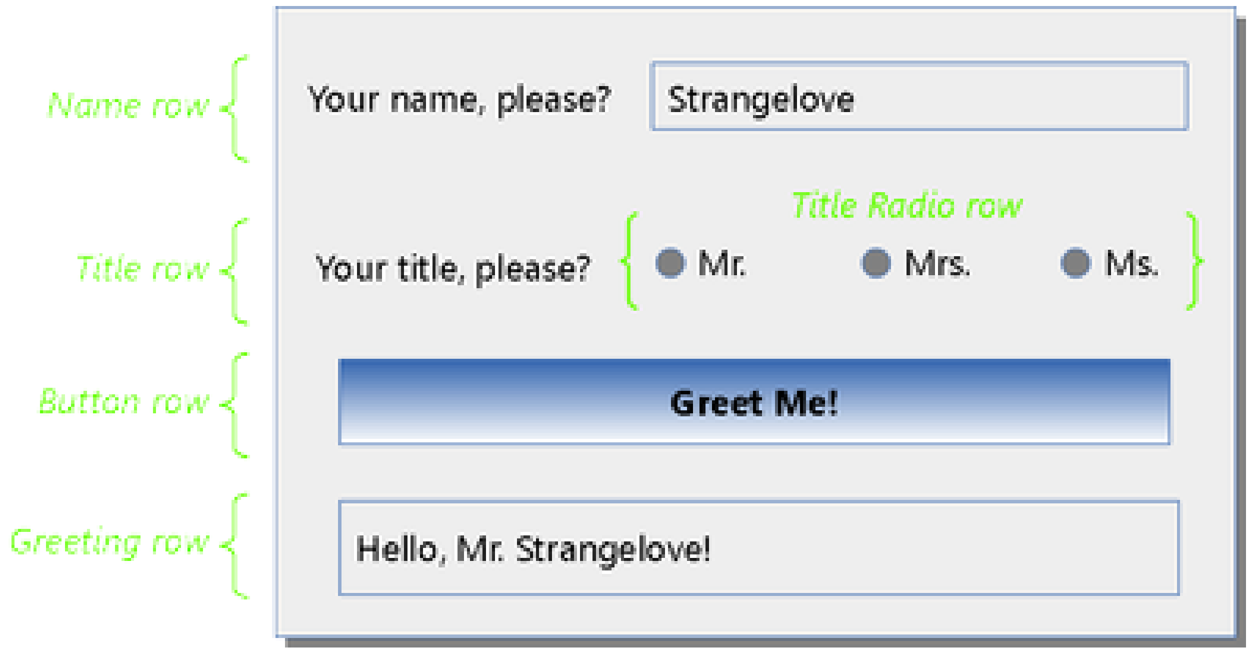

Figure 3 – Breakdown by rows – Title radio buttons

So it seems we have all the information we need. Just one minor thing: The overall window layout is a perfect fit for a SpecColumnLayout as 4 rows are vertically added next to each other.

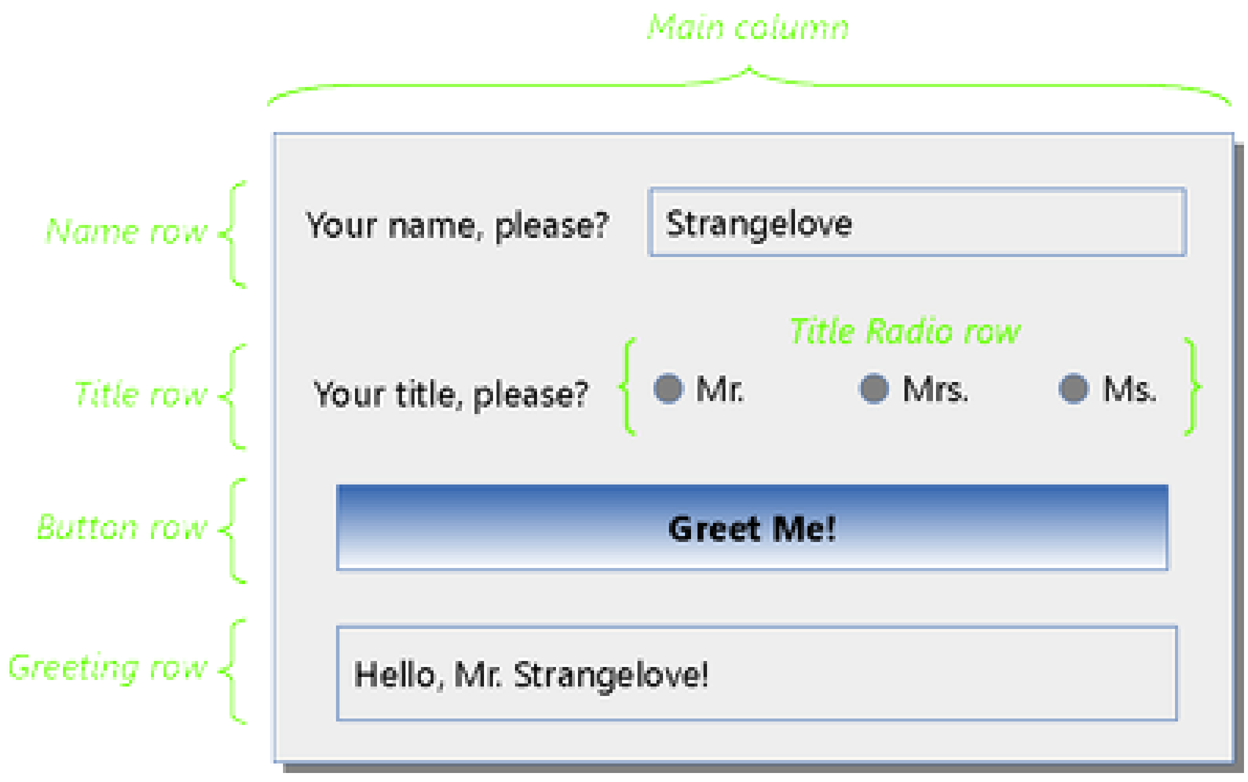

Figure 4 – Breakdown – Main column

Finally we have the design breakdown using our tools.

Pharo The Design!

Time to convert our design into something Pharo understands.

As the first step, we need to add two (informational) labels to our widgets.

```
ComposableModel subclass: #MyFirstWindow
	instanceVariableNames: 'labelGreeting textName buttonGreet
	classVariableNames: ''
	poolDictionaries: ''
	category: 'My-Spec-Tutorial'
```

Don't forget to **generate the accessors**.

As usual, the next step is to instantiate the widgets:

```
initializeWidgets
	self instantiateModels: #(
		labelGreeting	LabelModel
		textName	TextInputFieldModel
		buttonGreet	ButtonModel
		radioMr	RadioButtonModel
 7                 radioMs         RadioButtonModel
 8                 radioMrs        RadioButtonModel
 9                 labelName       LabelModel
10                 labelTitle      LabelModel
11         ).
12
13         labelGreeting text: ''.
14         textName autoAccept: true.
15         buttonGreet label: 'Greet Me!'; disable.
16         self setupTitleRadioButtons.
17         labelName text: 'Your name, please?'.
18         labelTitle text: 'Your title, please?'.
```

Note new lines 9, 10, 17 and 18.

And finally we have to tell Spec about the layout. As you can remember, modifying layout is done in the class-side `defaultSpec`:

```
defaultSpec
    ^ SpecLayout composed
        "Adding Main column"
        newColumn: [ :mainColumn |

            mainColumn
                "Adding Name row"
                newRow: [ :rowName |
                    rowName
                        add: #labelName;
                        add: #textName ].

            mainColumn
                "Adding Title row"
                newRow: [ :rowTitle |
                    rowTitle
                        add: #labelTitle;
                        "Adding Title Radio row"
                        newRow: [ :rowTitleRadio |
                            rowTitleRadio
                                add: #radioMr;
                                add: #radioMrs;
                                add: #radioMs ] ].

            mainColumn
                "Adding Button row"
                newRow: [ :rowButton |
                    rowButton
                        add: #buttonGreet ].
30
31          mainColumn
32              "Adding Greeting row"
33              newRow: [ :rowGreeting |
34                  rowGreeting
35                      add: #labelGreeting ] ];
36      yourself
```

On line 2, Spec is told to use `SpecLayout` for the layout of the window; `SpecLayout` is used whenever you have composite/nested layouts like the greeter's. On lines 8, 15, 27 and 33 rows for *Name*, *Title*, *Button* and *Greeting* are created. On line 19, the inner row for *Title Radio* is created.

One important thing to note is that when nesting a layout inside another layout using either `newRow:` or `newColumn:`, it should be done inside a block closure.

Run `MyFirstWindow new openWithSpec` to feast your eyes on your new design!

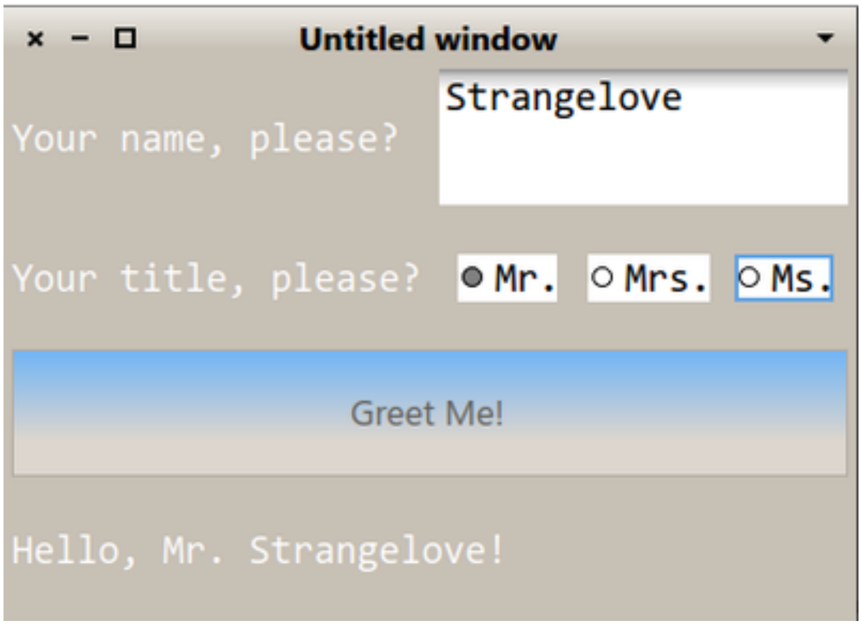

Figure 5 – Greeter with layouts

What Next?

We've discussed enough for a single episode. It's time that you play with layouts and get acquainted with how they work. As for the greeter, there are still rough edges to fix, like the window title and size or widget height/width. These topics will be covered in the next episode.

Please subscribe to this RSS feed to be notified when a new episode is available.

Filed under: spec, ui, pharo, smalltalk

Site Map
Accessibility
Contact