Pharo, Spec and GTK

(revisiting the desktop world)

About me

Esteban Lorenzano (@estebanlm)

- Pharo consortium engineer since 2018
- Pharo architect since 2012
- Owned a company to develop in Pharo back in 2008
- Java senior architect for 7 years (and 15 years overall java experience)
- Web, microprocessors, etc., etc., etc.
- JavaScript, C++, ObjC, C#, Delphi, ASM and lots of languages that no longer exist or have been long-time forgotten
- 26 years (!) programming experience

"How to do a desktop application with Pharo?"

– Most requested feature every ESUG (personal survey)

Last year the people of Schmidt called us…

Do you remember this?

2010

Mars

Just another world

Desktop applications?

- Desktop applications are going to be around for a while.
  - And until now we do not have a good solution (and yes, we have real requirements).

- Pharo itself is a desktop application.

What is Spec?

A set of widget presenters to build window components.

Why Spec?

- Modular design.
- Testable!
- Spec can be extended to cover what other frameworks do.
- Morphic is too low-level for regular applications.

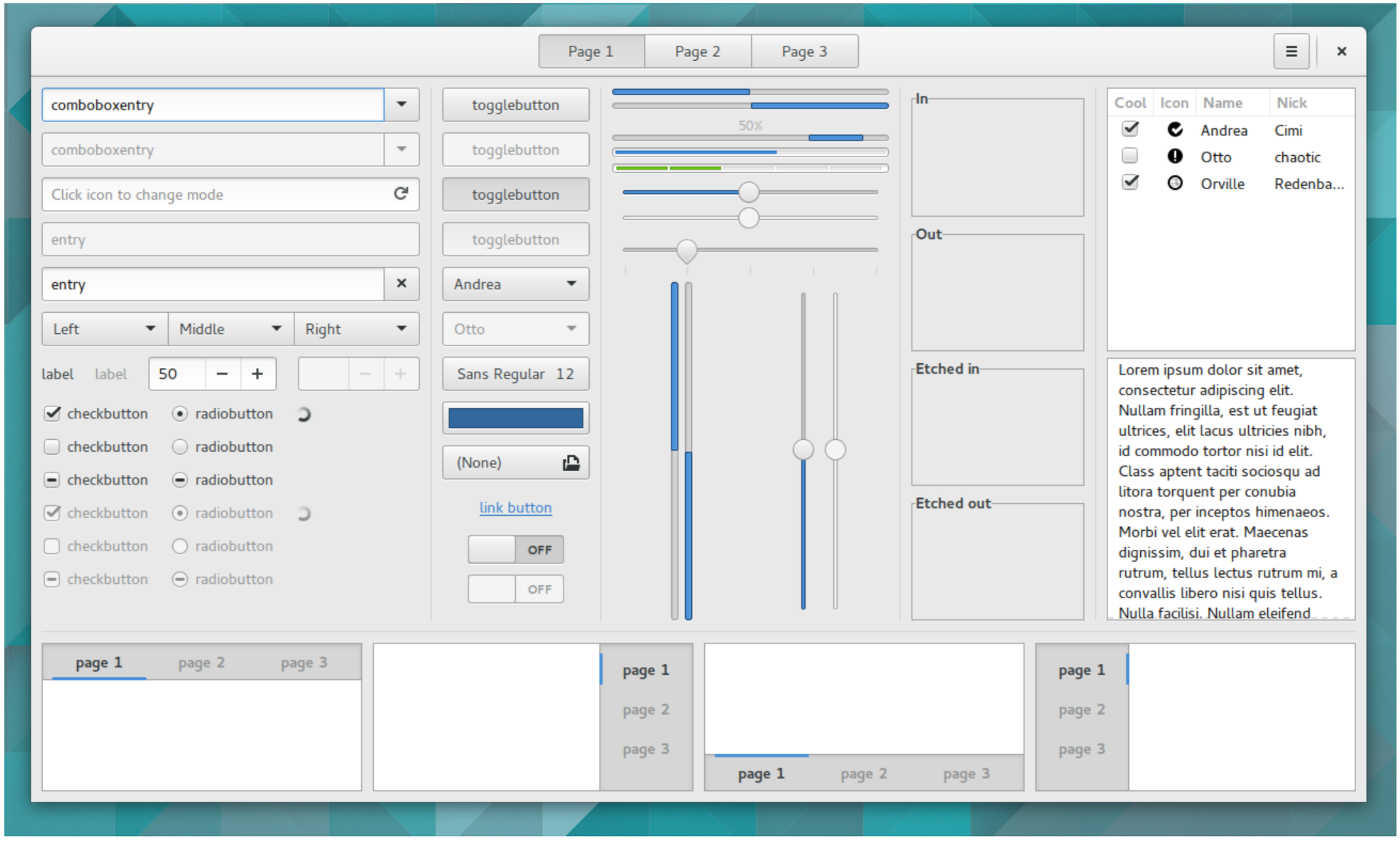
What is Spec 2.0?

- A revisit of the concepts of original Spec.
- Plus what we have learn.
- Plus what we like from other frameworks.
- Plus the possibility to plug different backends.

Spec 2.0 backends

- Morphic (to keep current world working)
- GTK+3
- In the future: bloc/brick

GTK+3


Page 1
Page 2
Page 3
comboboxentry
comboboxentry
Click icon to change mode
entry
entry
Left
Middle
Right
label
label
50
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
checkbutton
radiobutton
togglebutton
togglebutton
togglebutton
togglebutton
Andrea
Otto
Sans Regular 12
(None)
link button
OFF
OFF
50%
In
Out
Etched in
Etched out
Cool
Icon
Name
Nick
Andrea
Cimi
Otto
chaotic
Orville
Redenba...
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam fringilla, est ut feugiat ultrices, elit lacus ultricies nibh, id commodo tortor nisi id elit. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Morbi vel elit erat. Maecenas dignissim, dui et pharetra rutrum, tellus lectus rutrum mi, a convallis libero nisi quis tellus. Nulla facilisi. Nullam eleifend
page 1
page 2
page 3
page 1
page 2
page 3
page 1
page 2
page 3
page 1
page 2
page 3

GTK+3

- Cross platform
- Open source (LGPL)
- Mature
- Popular
- C bindings

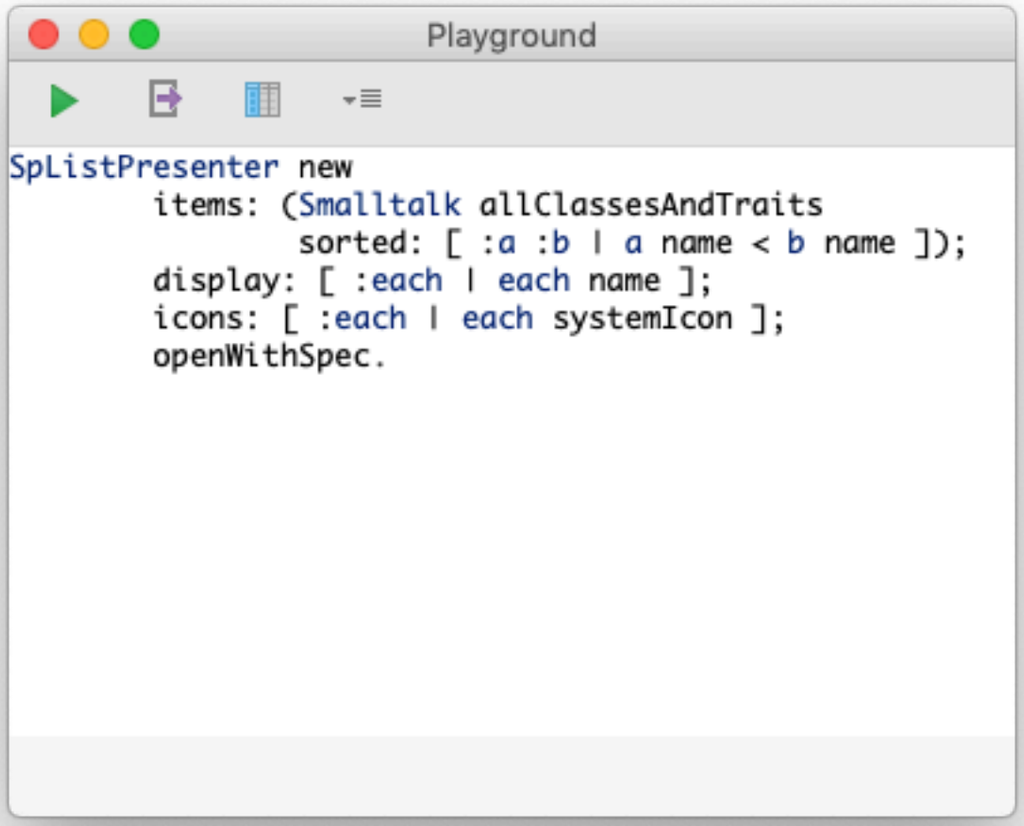
In Spec 2.0, there are a few new concepts around:

Presenters,
Applications,
Layouts,
Transmissions

Let's take a quick tour!

Presenters

- A presenter is an “atom”
  - There are widget presenters.
  - There are composed presenters (this is what users do most of the time).
- A presenter is always a high-level UI element. E.g. Not “a rectangle” but “a text area”.


Playground
SpListPresenter new
items: (Smalltalk allClassesAndTraits
sorted: [:a :b | a name < b name]);
display: [:each | each name];
icons: [:each | each systemIcon];
openWithSpec.

Application

- Starting point of a Spec 2.0 application.
- The place for UI resources (style, icons, etc.)
  - Perhaps the place to access application model (DB resources, etc).
- The place to configure your application before launch.
- … override **#start** to initiate your application.

Playground

```
| app |

app := SpApplication new run.

(app new: SpListPresenter)
        items: (Smalltalk allClassesAndTraits
                sorted: [ :a :b | a name < b name ]);
        display: [ :each | each name ];
        icons: [ :each | each systemIcon ];
        openWithSpec.
```

List

- AColorSelectorMorph
- ADPCMCodec
- AIFFFileReader
- ASTCache
- ASTCacheReset
- ASTCacheResetTest
- ASTClassBuilderTest
- ASTEvaluationTest
- ASTPluginMeaningOfLife
- ASTTransformExamplePluginActive
- ASTTransformationPluginTest
- Abort
- AboutDialogWindow

Playground

```
| app |

app := SpApplication new run.
app useBackend: #Gtk.

(app new: SpListPresenter)
        items: (Smalltalk allClassesAndTraits
                sorted: [ :a :b | a name < b name ]);
        display: [ :each | each name ];
        icons: [ :each | each systemIcon ];
        openWithSpec.
```

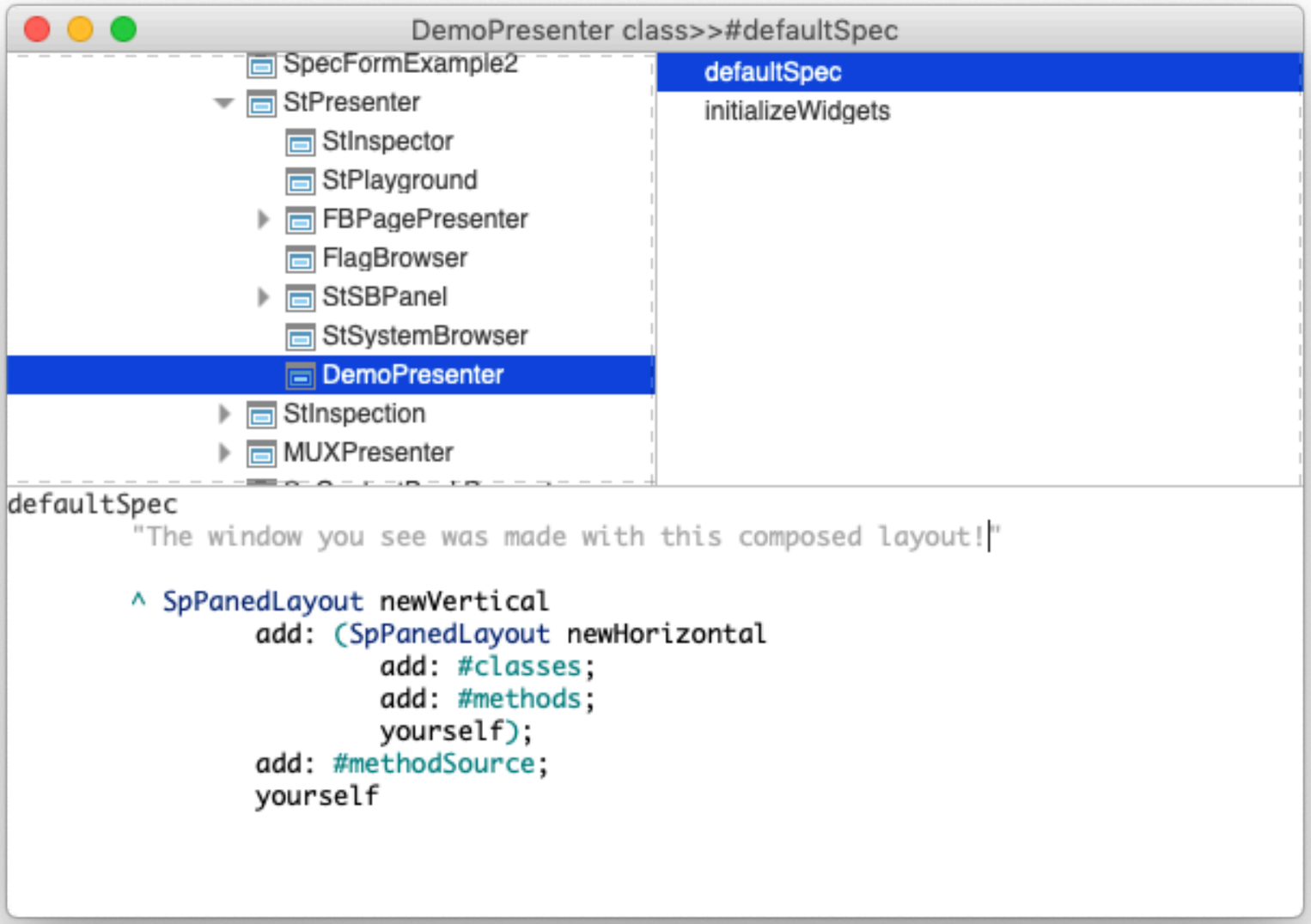
List

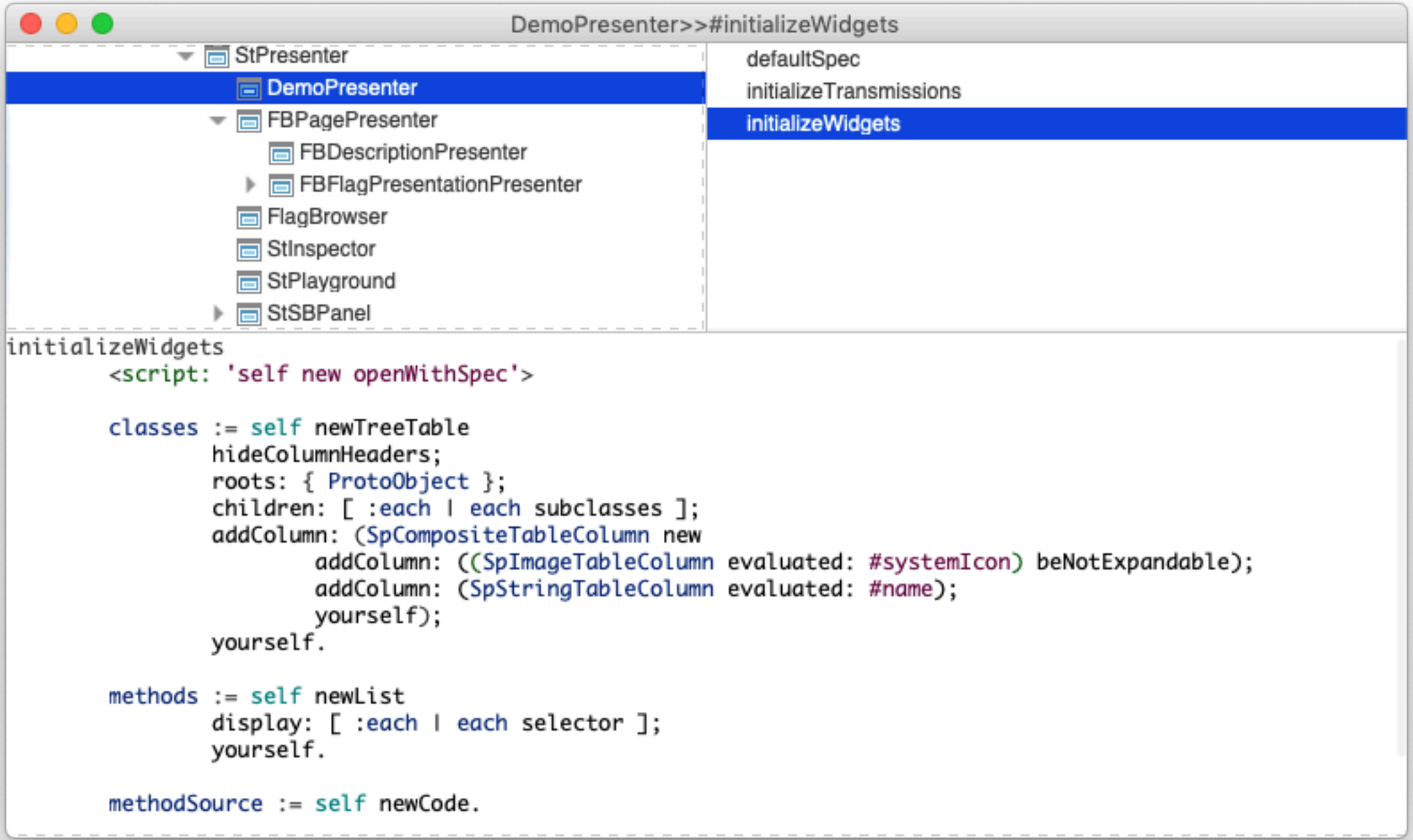
- ObjCStructure
- Object
- ObjectFinalizer
- ObjectFinalizerCollection
- ObjectFinalizerTest
- ObjectLayout
- ObjectMockForTest
- ObjectStringConverter
- ObjectTest
- ObjectsAsMethodsExample
- ObsoleteTest
- OkCancelToolbar
- OkToolbar
- OldSendsDeprecatedMethodToGlobalRule
- OmAbstractReference
- OmBlock
- OmBlockFileStore

Layouts, layouts, layouts.

- A layout defines how the presenters will be placed in my component.

- Instead one layout that does all, we have several for different tasks: Box, Grid, etc.


DemoPresenter class>>#defaultSpec
SpecFormExample2
StPresenter
StInspector
StPlayground
FBPagePresenter
FlagBrowser
StSBPanel
StSystemBrowser
DemoPresenter
StInspection
MUXPresenter
defaultSpec
initializeWidgets
defaultSpec
"The window you see was made with this composed layout!"
^ SpPanedLayout newVertical
add: (SpPanedLayout newHorizontal
add: #classes;
add: #methods;
yourself);
add: #methodSource;
yourself


DemoPresenter>>#initializeWidgets
StPresenter
DemoPresenter
FBPagePresenter
FBDescriptionPresenter
FBFlagPresentationPresenter
FlagBrowser
StInspector
StPlayground
StSBPanel
defaultSpec
initializeTransmissions
initializeWidgets
initializeWidgets
<script: 'self new openWithSpec'>
classes := self newTreeTable
hideColumnHeaders;
roots: { ProtoObject };
children: [:each | each subclasses];
addColumn: (SpCompositeTableColumn new
addColumn: ((SpImageTableColumn evaluated: #systemIcon) beNotExpandable);
addColumn: (SpStringTableColumn evaluated: #name);
yourself);
yourself.
methods := self newList
display: [:each | each selector];
yourself.
methodSource := self newCode.

Connecting presenters

- Transmissions was a great idea from Glamour.

- You plug an output port of presenter A to input port of presenter B.

DemoPresenter>>#initializeTransmissions

StPresenter
DemoPresenter
FBPagePresenter
FBDescriptionPresenter
FBFlagPresentationPresenter
FlagBrowser
StInspector
StPlayground
StSBPanel

defaultSpec
initializeTransmissions
initializeWidgets

initializeTransmissions

```
	"transmit from classes to methods"
	classes transmitDo: [ :what | self window title: what gtDisplayString ].
	(classes transmitTo: methods)
		preTransmission: [ :to :from :what | self window title: what name ];
		transform: [ :each |
			(each class methods sorted: #selector ascending),
			each methods sorted: #selector ascending ].

	"transmit from methods to methodSource (with behaviour)"
	methods transmitDo: [ :what | self window title: what gtDisplayString ].
	(methods transmitTo: methodSource)
		preTransmission: [ :to :from :what | to behavior: what methodClass ];
		transform: [ :each | each sourceCode ].
```

The Pharo IDE

Last year the people of Lifeware called us…

The Pharo IDE

- By Pharo 9, all tools will be made with Spec 2.0.
  - This will allow us to switch backends (and finally decommission Morphic).
- It will be possible to execute Pharo in “GTK mode”.

Playground

```
42 factorial.
"140500611775287989854314260624

StInspector openOn: Morph new.
```

bounds:

MonticelloTonel-FileSyste
MonticelloTonel-Tests
Moose-Algos-Graph
MoreColors
Morphic-Base
Morphic-Core
Announcements
Borders
Events
Extensions
Kernel

BorderedMorph
HandMorph
MorphExtension
Morph

Instance | Class | Variables

instance side
halos and balloon help
rounding
viewer
thumbnail
events-alarms
testing
recategorized
card in a stack
button

bottomCenter
bottomLeft
bottomLeft:
bottomRight
bottomRight:
boundingBoxOfSubmorphs
bounds
bounds:
bounds:from:
bounds:in:
boundsForBalloon

Comment | Morph | bounds:

```
bounds: newBounds
	| oldExtent newExtent |
	oldExtent := self extent.
	newExtent := newBounds extent.
	(oldExtent dotProduct: oldExtent) <= (newExtent dotProduct: newExtent) ifTrue:[
		"We're growing. First move then resize."
		self position: newBounds topLeft; extent: newExtent.
	] ifFalse:[
		"We're shrinking. First resize then move."
		self extent: newExtent; position: newBounds topLeft.
	].
```

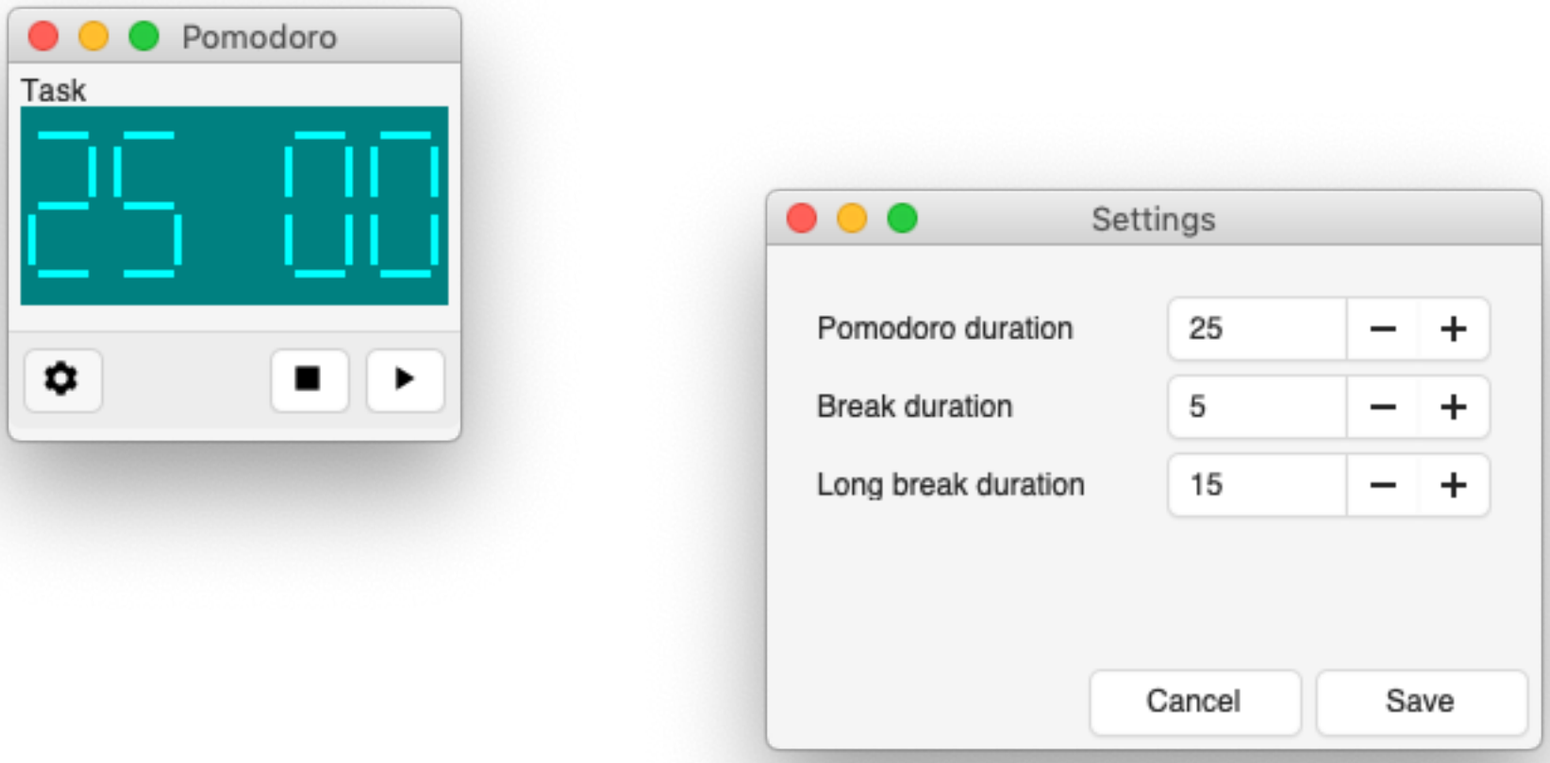
Inspector on a

Raw | Morph | Breakpoints | Meta

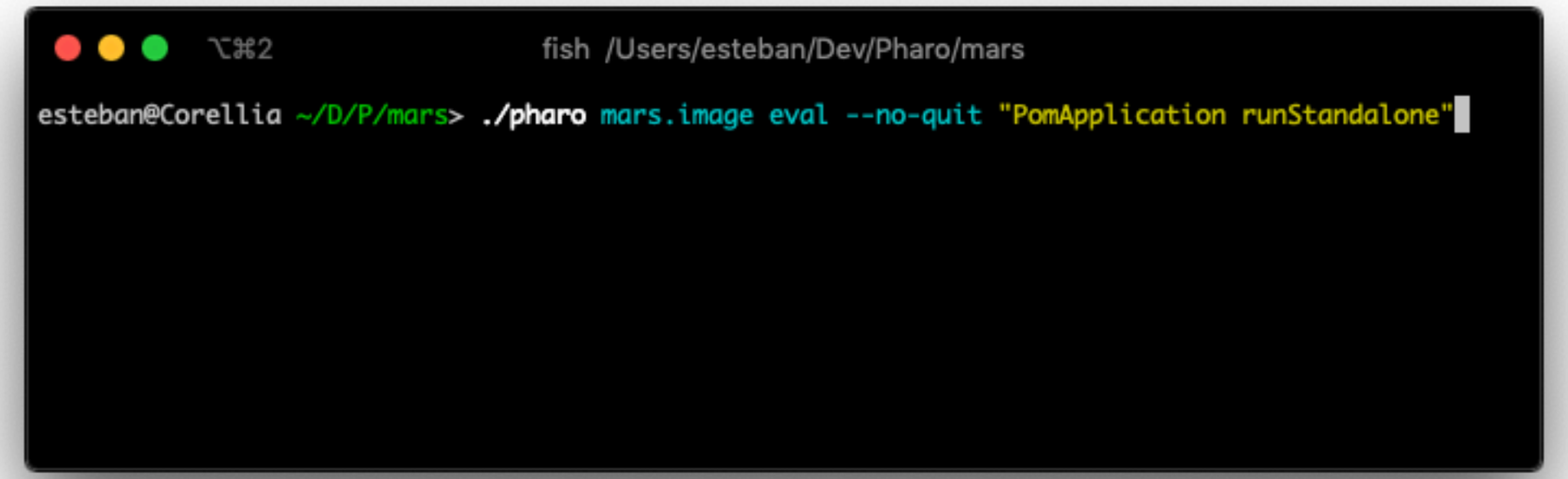
Variable	Value
self	a Morph(130220544)
bounds	(0@0) corner: (50@4
owner	nil
submorphs	an Array [0 items] ()
fullBounds	nil
color	Color blue
extension	nil

(Some prototypes)

Finally, a desktop application



How to run standalone?



But of course, this is fine for development…

For production, you will be able to install your application as “default application”, then Pharo will run it instead the IDE.

Spec 2.0

- GTK+3 and Morphic backends now, other(s) in the future.
- Standalone applications.
- More and better widgets.
- Styles.
- Better layouts.
- Transmissions.

Try it now!
(just remember it is still alpha :)
https://github.com/pharo-spec/mars-gtk
(and follow the instructions)