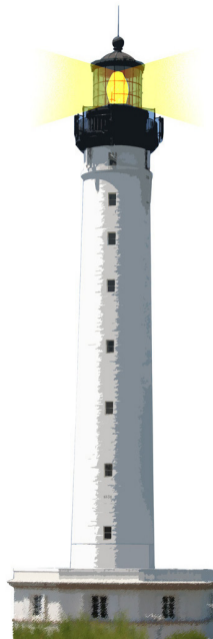


# Delegation of actions and accumulator

Form validation as an example

S. Ducasse



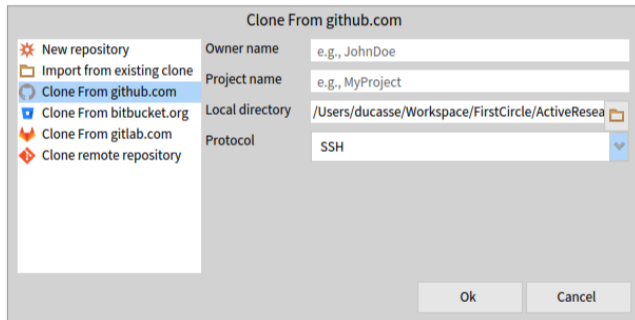
# Scenario

- How can we navigate a tree of instances (widgets)?
- Where children can decide to be skipped?





# The case: Validation

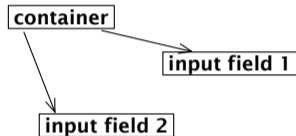
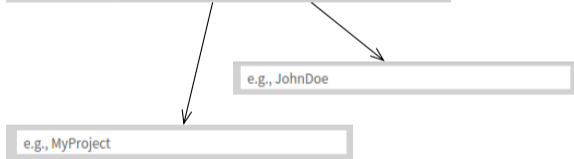
- We want to validate UI forms
- Nested components may want to validate **or not** their contents
  - at input field or just at the OK level



The image shows a dialog box titled "Clone From github.com". On the left, there is a list of options: "New repository", "Import from existing clone", "Clone From github.com" (highlighted), "Clone From bitbucket.org", "Clone From gitlab.com", and "Clone remote repository". On the right, there are four form fields: "Owner name" with the placeholder "e.g., JohnDoe", "Project name" with the placeholder "e.g., MyProject", "Local directory" with the path "/Users/ducasse/Workspace/FirstCircle/ActiveResea" and a folder icon, and "Protocol" with the value "SSH" and a dropdown arrow. At the bottom right, there are "Ok" and "Cancel" buttons.

# A tree of instances

Owner name	e.g., JohnDoe
Project name	e.g., MyProject
Local directory	/Users/ducasse/Workspace/FirstCircle/ActiveResea 
Protocol	SSH 



# A first design

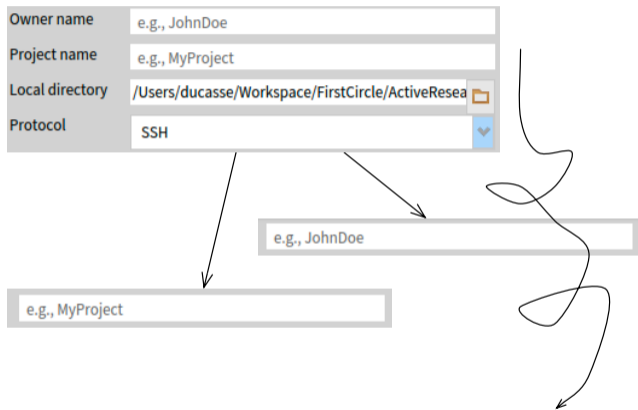
- Any presenter can validate its contents
- Per default does not nothing

```
SpPresenter >> validate  
  ^ self
```

```
SpOptionPresenter >> validate
```

```
| report |  
report := SpValidationReport new.  
self children do: [ :presenter | presenter validate ]
```

# Flow's first design

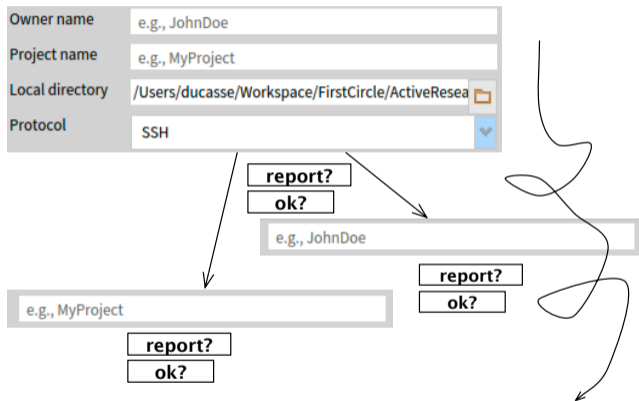


# Analysis

- We need to have a report to know if the validation failed or not
- Should `validate` return a report?
- If `validate` returns a report then we have to return an ok report for anybody
- Force a report for any instances?



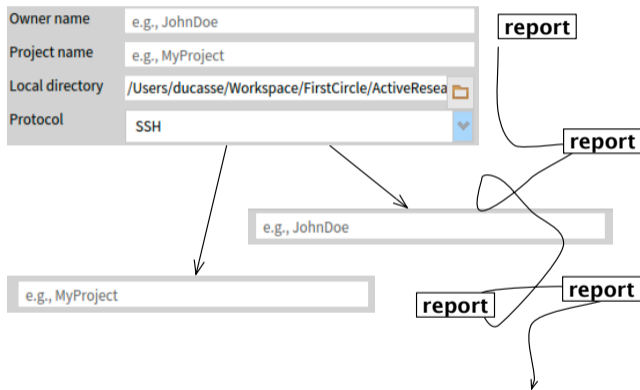
# Flow's first design





# Second design: provide an accumulator

Pass around a basket and let any sub instance decides if it wants to participate



## Second design: code

```
SpPresenter >> validateInto: aReport  
  ^ self
```

```
SpOptionPresenter >> validate
```

```
| report |  
report := SpValidationReport new.  
self children do: [ :presenter | presenter validateInto: report ].  
^ report
```

# Local and global together

```
SpTextFieldWithValidation >> validateInto: aValidationReport  
  self validate.  
  aValidationReport addAll: validationErrors
```

Each validating subcomponent

- gets the responsibility to fill up the report
- can bring its information to the report



# Conclusion

- Let the object decides if it wants to join a process but passing a container



A course by

S. Ducasse, G. Polito, and Pablo Tesone



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France  
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>