

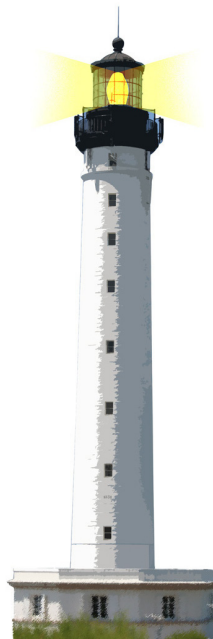


Advanced Object-Oriented Design Lecture

About Validation



<http://www.pharo.org>



Goal

- How can we navigate an instance tree?
- With optional states that can be skipped



The case

- We want to validate UI forms
- Nested components may want to validate or not their contents

Screenshot here

A tree of instances

Screenshot here



Validate First Design

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

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

```
SpOptionPresenter >> validate
```

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



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
- We do not want to force a report for all the tree instances...

Second Design: let the object decide

Pass around a basket and let any sub instance decide if it want to participate

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

```
SpOptionPresenter >> validate  
  "Return a "  
  | report |  
  report := SpValidationReport new.  
  self layout children do: [ :presenter | presenter validateInto: report ].  
  ^ report
```

Bringing Local and Global

- Each validating subcomponents can bring its information to the report.

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



Conclusion

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



A course by

S. Ducasse, L. Fabresse, 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/>