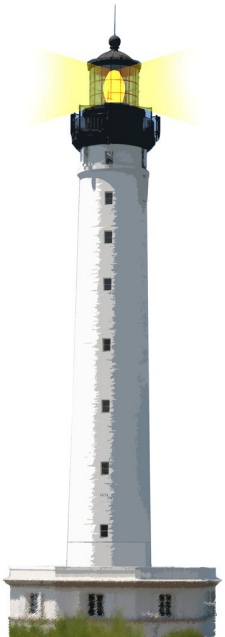
Reflection: Stack as an Object

Damien Cassou, Stéphane Ducasse and Luc Fabresse

W7S06

Pharo

http://www.pharo.org



Just to Reveal a Bit of It

To let you know that it exists

- On demand the stack can be turned into an object
- We can walk but also **modify** it

Coding in the Debugger

- When a method is not found, we may end up with a debugger
- In the debugger, we can ask for the creation of a method on the fly
- The system compiles on the spot a special method
- Then it reexecutes the method
- It raises a shouldBeImplemented exception
- Then you can edit the method in the debugger
- Then proceed and the program continues to run

Operation Supporting Coding in the Debugger

- On the fly method definition
- Stack navigation / manipulation

Stack Frame as an Object

- Execution stack can be turned into a live object
- By default the stack is not an object
  - Can be reified on demand
- Support exception definitions from within the language
- Basis for continuations and web serving

thisContext

- thisContext is one of the three pseudovariables with self and super
- Returns an object that represents the method activation
- Can walk and change the stack
- Put self halt in the code to see it and walk

Example: Better Deprecated Message

We declare deprecation

```
A >> foo
  self deprecated: 'Use bar.' on: '31/12/2015' in: 'Pharo50'.
  self bar
```

should print: 'Message foo is deprecated in Pharo50. Use bar'

Example: Better Deprecated Message

```
deprecated: anExplanationString on: date in: version
  "Warn that the sending method has been deprecated"

  (Deprecation
    method: thisContext sender method
    explanation: anExplanationString
    on: date
    in: version) signal
```

thisContext sender method **returns compiled method** A>>#foo

Example: Conditional Halt and Test

- How to halt a method that is heavily used?
- How to stop execution only from a given execution flow?

```
foo
  ...
  Halt if: #testSetInitialized
  ...
```

Only halt if executed from testSetInitialized

if: Implementation

```
Halt class >> if: condition
 "This is the typical message to use for inserting breakpoints
     during debugging.
 The argument can be one of the following:
  – a block: if the Block has one arg, the calling object is bound
     to that.
  – an expression
  – a selector: Halt if found in the call chain"

 condition isSymbol ifTrue: [ ^ self haltIfCallChainContains:
     condition ].
 condition isBlock ifTrue: [ ^ self haltIfBlockWithCallingObject:
     condition].
 condition ifTrue: [self signal].
```

How if: is implemented?

```
Halt class >> haltIfCallChainContains: aSelector
  | cntxt |
  cntxt := thisContext.
  [ cntxt sender isNil ] whileFalse: [
    cntxt := cntxt sender.
    (cntxt selector = aSelector) ifTrue: [ self signal ] ].
```

Foundation for Innovation

- Continuations
- Seaside: Powerful dynamic web framework for dynamic web applications
  - http://www.seaside.st
  - http://book.seaside.st

A course by

and

in collaboration with