

Playing it Safe

Tips for Safe Programming Practices for Ruby

Jim Weirich
Chief Scientist
EdgeCase LLC



Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

1



Contact:

jim.weirich@gmail.com
<http://onestepback.org>
[@jimweirich](#)

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

2

Gino Tesei Asks ...

Are Ruby's Open Classes a Poor Fit for Large Projects?

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

3

```
class Folder
  # ...
  def list
    puts @name+":"
    for i in 0...@files.length
      puts "--"+@files[i]
    end
  end
  # ...
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

4

```
#base semantics
aFolder = Folder.new("reports")
aFolder.list
puts "here we're ..."
```

```
reports:
here we're ...
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

5

```
class Folder
  # ...
  def list
    puts @name+": "
    if @files.length == 0
      raise "empty folder!"
    end
    for i in 0...@files.length
      puts "--"+@files[i]
    end
  end
  # ...
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

6

```
#base semantics
aFolder = Folder.new("reports")
aFolder.list
puts "here we're ..."
```

```
reports:
folder2.rb:14:in `list': empty folder!
(RuntimeError)
  from folder2.rb:25
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

7

Zed Shaw Rants ...

The Chainsaw Infanticide Logger Maneuver

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

8

“One of the things
that's really great
about agile languages is
they give you the
power to do
anything. ...”

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

9

“... One of the most horrible
things about agile languages is
they give every other idiot the
same power to stab you in the
back with a rusty pitchfork.”

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

10

“Not only does this code
re-open a class to do
something that could
**just as easily be
done with
subclassing, ...**”

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

11

“...but the author also
TURNS OFF
DOCUMENTATION
with :nodoc: so that nobody
knows about it. ...”

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

12

**“...This is more like
Principle Of Most
Heinous Arsenic
Injection.”**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

13

**“I then trolled through a
few more projects and
found that this is
common practice.”**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

14

```
class Logger

  SIMPLE_FORMAT = "%5s: %s\n"
  @@global_logger = Logger.new(STDERR)

  def self.set(logger)
    @@global_logger = logger
  end

  def self.get
    @@global_logger
  end

  def self.debug(str)
    @@global_logger.debug(str)
  end
  #...
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

15

Logger.warn "Munged Logger"

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

16


```
Log = Logger.new(STDOUT)
...
Log.warn "Standard Logger"
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

17

```
#--
# THINK: some people think,
# this extension is dangerous,
# investigate.
#++
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

18

Does it really matter?

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

19

Application vs Library

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

20

A Metaphor

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

21

“At least I have chicken!”

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

22

Tips for Playing it Safe

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

23

How to Avoid Being a Leeroy Jenkins

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

24

Guard against Murphy, not Machiavelli

-- Damian Conway

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

25

Applications vs Libraries

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

26

Tip

Use NameSpaces

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

27

How Many?

```
class Node  
  ...  
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

28

```
IRB::Node
REXML::Light::Node
Tk::BLT::Tree::Node
Tk::BLT::Treeview::Node
Tk::BWidget::Tree::Node
NQXML::Node
YAML::Syck::Node
HTML::Node
DispatcherTest::Node
Node
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

29

```
IRB::Node
REXML::Light::Node
Tk::BLT::Tree::Node
Tk::BLT::Treeview::Node
Tk::BWidget::Tree::Node
NQXML::Node
YAML::Syck::Node
HTML::Node
DispatcherTest::Node
Node
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

30

```
IRB::Node
REXML::Light::Node
Tk::BLT::Tree::Node
Tk::BLT::Treeview::Node
Tk::BWidget::Tree::Node
NQXML::Node
YAML::Syck::Node
HTML::Node
DispatcherTest::Node
Node
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

31

Corollary

Choose
Project Name
Carefully

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

32

- RubyForge
- gems
- github

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

33

Tip

Avoid Top Level
Functions and
Constants

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

34

Rake

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

35

```
class Task ... end
class RakeApp ... end
class FileList ... end

def task ... end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

36

```
module Rake
  class Task ... end
  class RakeApp ... end
  class FileList ... end
end

def task ... end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

37

```
module Rake
  class Task ... end
  class RakeApp ... end
  class FileList ... end
end

def task ... end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

38

```
module Rake
  class Task ... end
  class RakeApp ... end
  class FileList ... end
end
FileList = Rake::FileList
def task ... end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

39

Tip

Avoid Modifying Existing Classes

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

40

Tip

Prefer Adding over Modifying

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

41

```
module Kernel
  def rake_dup()
    dup
  end
end

[
  NilClass, FalseClass, TrueClass,
  Fixnum, Symbol
].each do |clazz|
  clazz.class_eval {
    def rake_dup() self end
  }
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

42

```

module Kernel
  def rake_dup()
    dup
  end
end

[
  NilClass, FalseClass, TrueClass,
  Fixnum, Symbol
].each do |clazz|
  clazz.class_eval {
    def rake_dup() self end
  }
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

43

```

module Kernel
  def rake_dup()
    dup
  end
end

[
  NilClass, FalseClass, TrueClass,
  Fixnum, Symbol
].each do |clazz|
  clazz.class_eval {
    def rake_dup() self end
  }
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

44

```
module Kernel
  def rake_dup()
    dup
  end
end

[
  NilClass, FalseClass, TrueClass,
  Fixnum, Symbol
].each do |clazz|
  clazz.class_eval {
    def rake_dup() self end
  }
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

45

Tip

When Adding,
use project
scoped names.

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

46

```
fl = FileList[
  "a.pl", "b.pl"
]
fl.ext("rb")
=> ["a.rb", "b.rb"]
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

47

```
fn = "a.pl"
fn.ext("rb")
=> "a.rb"
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

48


```
class String
  unless instance_methods.include? "ext"
    def ext(newext='')
      ...
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

49

```
class String
  unless instance_methods.include? "ext"
    def ext(newext='')
      ...
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

50

```
class String
  unless instance_methods.include? "ext"
    def ext(newext='')
      ...
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

51

Tip

When Adding
Public Methods,
ask first.

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

52

Selector Namespaces

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

53

Replacing Behavior

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

54

```
module Rake
  class Task ... end
  class RakeApp ... end
  class FileList ... end
end
FileList = Rake::FileList
def task ... end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

55

```
module Rake
  class Task ... end
  class RakeApp ... end
  class FileList ... end
end
FileList = Rake::FileList
def task ... end
if classic_namespace?
  Task = Rake::Task
  RakeApp = Rake::RakeApp
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

56

A Cool Trick

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

57

```
class Module
  alias :rake_original_const_missing :const_missing
  def const_missing(const_name)
    case const_name
    when :Task
      Rake.application.const_warning(const_name)
      Rake::Task
    when :FileTask
      Rake.application.const_warning(const_name)
      Rake::FileTask
    ...
    else
      rake_original_const_missing(const_name)
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

58

```

class Module
  alias :rake_original_const_missing :const_missing
  def const_missing(const_name)
    case const_name
    when :Task
      Rake.application.const_warning(const_name)
      Rake::Task
    when :FileTask
      Rake.application.const_warning(const_name)
      Rake::FileTask
    ...
    else
      rake_original_const_missing(const_name)
    end
  end
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

59

```

class Module
  alias :rake_original_const_missing :const_missing
  def const_missing(const_name)
    case const_name
    when :Task
      Rake.application.const_warning(const_name)
      Rake::Task
    when :FileTask
      Rake.application.const_warning(const_name)
      Rake::FileTask
    ...
    else
      rake_original_const_missing(const_name)
    end
  end
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

60

Hook Tip

Chain to the Next Hook

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

61

```
class Module
  alias :rake_original_const_missing :const_missing
  def const_missing(const_name)
    case const_name
    when :Task
      Rake.application.const_warning(const_name)
      Rake::Task
    when :FileTask
      Rake.application.const_warning(const_name)
      Rake::FileTask
    ...
    else
      rake_original_const_missing(const_name)
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

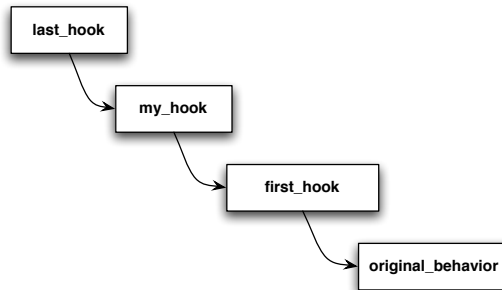
62

```
class Module
  alias :rake_original_const_missing :const_missing
  def const_missing(const_name)
    case const_name
    when :Task
      Rake.application.const_warning(const_name)
      Rake::Task
    when :FileTask
      Rake.application.const_warning(const_name)
      Rake::FileTask
    ...
  else
    rake_original_const_missing(const_name)
  end
end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

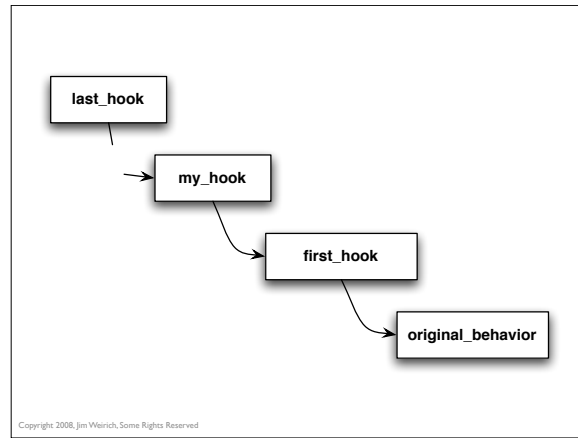
63



Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

64



Wednesday, August 18, 2010

65

Hook Tip

Only handle
Your special
cases

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

66

```
class Module
  alias :rake_original_const_missing :const_missing
  def const_missing(const_name)
    case const_name
    when :Task
      Rake.application.const_warning(const_name)
      Rake::Task
    when :FileTask
      Rake.application.const_warning(const_name)
      Rake::FileTask
    ...
    else
      rake_original_const_missing(const_name)
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

67

Tip

Limit the scope of
your hook

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

68

```
thing = db.fetch_data(key)
thing.do_something unless thing.nil?
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

69

```
thing = db.fetch_data(key)
thing.do_something unless thing.nil?
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

70

```
thing = db.fetch_data(key)
thing.do_something
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

71

```
class NilClass
  def method_missing(sym, *args, &block)
    nil
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

72

```
module Kirby
  class Null
    def method_missing(sym, *args, &block)
      nil
    end
  end
end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

73

Whiny Nils

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

74

```
class NilClass
  def method_missing(method, *args, &block)
    raise_nil_warning_for(...)
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

75

```
class NilClass
  def raise_nil_warning_for(
    klass = nil,
    selector = nil,
    with_caller = nil)
    message = "...nil object when you didn't expect it!"
    message << "\n... expected ... #{klass}." if klass
    message << "\n...while ... nil.#{selector}" if selector

    raise NoMethodError, message, with_caller || caller
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

76

Why are Whiny Nils OK?

Wednesday, August 18, 2010

77

Require Hooks

- RubyGems
- MockFS
- ActiveSupport (Rails)
- urirequire

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

78

```

module Kernel
  alias gem_original_require require
  def require(path)
    gem_original_require path
  rescue LoadError => load_error
  begin
    @gempath_searcher ||= Gem::GemPathSearcher.new
    if spec = @gempath_searcher.find(path)
      Gem.activate(spec.name, ...)
      gem_original_require path
    else
      raise load_error
    end
  end
end
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

79

```

module Kernel
  alias gem_original_require require
  def require(path)
    gem_original_require path
  rescue LoadError => load_error
  begin
    @gempath_searcher ||= Gem::GemPathSearcher.new
    if spec = @gempath_searcher.find(path)
      Gem.activate(spec.name, ...)
      gem_original_require path
    else
      raise load_error
    end
  end
end
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

80


```

module Kernel
  alias gem_original_require require
  def require(path)
    gem_original_require path
  rescue LoadError => load_error
  begin
    @gempath_searcher ||= Gem::GemPathSearcher.new
    if spec = @gempath_searcher.find(path)
      Gem.activate(spec.name, ...)
      gem_original_require path
    else
      raise load_error
    end
  end
end
end
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

81

```

module Kernel
  alias gem_original_require require
  def require(path)
    gem_original_require path
  rescue LoadError => load_error
  begin
    @gempath_searcher ||= Gem::GemPathSearcher.new
    if spec = @gempath_searcher.find(path)
      Gem.activate(spec.name, ...)
      gem_original_require path
    else
      raise load_error
    end
  end
end
end
end

```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

82

```
module Kernel
  alias gem_original_require require
  def require(path)
    gem_original_require path
  rescue LoadError => load_error
  begin
    @gempath_searcher ||= Gem::GemPathSearcher.new
    if spec = @gempath_searcher.find(path)
      Gem.activate(spec.name, ...)
      gem_original_require path
    else
      raise load_error
    end
  end
end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

83

But ...

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

84

```
$ irb
irb(main):001:0> require 'mylib'
=> true
irb(main):002:0> require 'mylib'
=> false
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

85

```
$ irb
irb(main):001:0> require 'mylib'
=> false
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

86

```
module Kernel
  alias gem_original_require require
  def require(path)
    gem_original_require path
  rescue LoadError => load_error
  begin
    @gempath_searcher ||= Gem::GemPathSearcher.new
    if spec = @gempath_searcher.find(path)
      Gem.activate(spec.name, ...)
      gem_original_require path
    else
      raise load_error
    end
  end
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

87

Tip

Preserve the
Original Behavior

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

88

Tip

Understand and Obey Contracts

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

89

Two Parts

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

90

What must be
true **before** a
method is called

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

91

What must be
true **before** a
method is called

precondition

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

91

What must be
true **after** a
method is called

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

92

What must be
true **after** a
method is called

postcondition

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

92

Math.sqrt(n)

Precondition

Postcondition

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

93

Math.sqrt(n)

Precondition

- $n \geq 0$

Postcondition

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

93

`Math.sqrt(n)`

Precondition

- `n >= 0`

Postcondition

- `(Result*Result - n).abs < 0.001`

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

93

New Behavior Must ...

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

94

New Behavior Must ...

- Have the Same or *Weaker* Precondition

`Math.sqrt(n)`

Precondition: **`n >= 0`**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

95

New Behavior Must ...

- Have the Same or *Weaker* Precondition

`Complex.sqrt(n)`

Precondition: **`true`**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

96

New Behavior Must ...

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

97

New Behavior Must ...

- Have the Same or *Stronger* Postcondition

`Math.sqrt(n)`

Postcondition: **`(result*result - n).abs < 0.001`**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

98

New Behavior Must ...

- Have the Same or *Stronger* Postcondition

`ReallyAccurate.sqrt(n)`

Postcondition: **`(result*result - n).abs < 0.000001`**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

99

Require no more.
Promise no less.

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

100

require filename

Precondition

file_in_loadpath?(filename)

Postcondition

previously_loaded? => Result == false

! previously_loaded? => Result == true

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

101

require filename

Precondition

file_in_loadpath?(filename)

Postcondition

previously_loaded? => Result == false

! previously_loaded? => Result == true

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

102

```
class Folder
  # ...
  def list
    puts @name+ ":"
    for i in 0...@files.length
      puts "--"+@files[i]
    end
  end
  # ...
end
```

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

103

Folder#list

(original)

Precondition

true

Postcondition

file_list_is_displayed

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

104

Folder#list

(munged)

Precondition

! @list.empty?

Postcondition

file_list_is_displayed

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

105

Folder#list

(munged)

Precondition

! @list.empty?

Postcondition

file_list_is_displayed

**Stronger
Precondition!**

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

105

Folder#list

(munged)

Precondition

! @list.empty?

Postcondition

file_list_is_displayed

Stronger

Precondition!

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

105

Replaced Behavior
Must be
“DUCK TYPE”
Compatible

Wednesday, August 18, 2010

106

Contracts Define “DUCK TYPE” Compatible

Wednesday, August 18, 2010

107

Summary

Wednesday, August 18, 2010

108

"What if someone
changes the logic of
+ for math
expressions. ..."

-- Dave Thomas

Wednesday, August 18, 2010

109

"... you take them out in
the parking lot and beat
them with a rubber hose!
The language shouldn't
prohibit us from doing
powerful things."

-- Dave Thomas

Wednesday, August 18, 2010

110

- Use Namespaces
- Choose Project Name Carefully
- Avoid Top Level Functions and Constants
- Avoid Modifying Existing Classes
- Prefer Adding over Modifying
- Use Project Scoped Names
- Ask Before Adding Public Methods
- Chain to the Next Hook
- Only Handle Your Special Cases
- Limit the Scope of your Hook
- Understand and Respect Contracts

Wednesday, August 18, 2010

111

Summary

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

112

Summary

Be Polite with Global Resources

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

112

Summary

Be Polite with Global Resources

Respect and Obey Contracts

Copyright 2008, Jim Weirich, Some Rights Reserved

Wednesday, August 18, 2010

112

License

This presentation is made available under the Creative Commons Attribution/Non-Commercial License, version 2.0. This means you are able to copy, distribute, display, and perform the work and to create derivative works, under the following conditions:

- You must give the original author credit.
- You may not use this work for commercial purposes.

(see <http://creativecommons.org/licenses/by-nc/2.0/> for details)

Copyright 2008, Jim Weirich, Some Rights Reserved