Hamburg on Ruby

Heimathafen der Hamburger Ruby Community

10. Februar 2016

http://hamburg.onruby.de/

HOSTED BY

SUM.CUMO

MADE TO INNOVATE

Andy Wenk
señor developer @ sum.cumo GmbH since 2014

best daddy award 2007 and 2010
musician (Metal rules u know!)
CouchDB guy
cyclist
massive book reader
@awenkhh

●●●

WE ARE HIRING

sum.cumo

MADE TO INNOVATE

Über uns

- Gründung 2010
- 24 Mitarbeiter (ab April 27)
- Ruby, Rails, JavaScript, CSS, HTML, Design, Sysadministration und vieles mehr
- Partner für Digitalisierung: Strategie, Planung, Technik, Kommunikation
- Kernkompetenz im Bereich Versicherungen und Lotterien
- Allianz, SwissRE, Dextra, Die Bayerische, LOTTO[.de | Niedersachsen | Hamburg]

Sonst so: **sssgeil hier!**

Ask us please ...

Eric

Moritz

Andy

Refinements - we no need Monkey's

•••

Hamburg On Ruby
10.02.2016

Monkey Patches?

What is a Monkey Patch?

Due to Ruby's open classes you can ***redefine*** *or* ***add functionality*** *to* ***existing classes****.*
This is called a "monkey patch". Unfortunately the ***scope*** *of such changes is* ***global****.*

All users of the monkey-patched class see the same changes. This can cause unintended side-effects or breakage of programs.

http://ruby-doc.org/core-2.3.0/doc/syntax/refinements_rdoc.html

In short ...

- redefine / add functionality to existing class
- scope is global
- every consumer will use the changes

global scope pollution !

code example

```
require 'json'

class Hash

  def size
    self.length * 2
  end

  def to_json
    JSON.generate(self)
  end

end
```

This is bad!

What are Refinements

Refinements are designed to **reduce** the **impact of monkey patching** on other users of the monkey-patched class. Refinements provide a way to **extend a class locally**.

http://ruby-doc.org/core-2.3.0/doc/syntax/refinements_rdoc.html

In short ...

“A mechanism to change the behaviour of an object in a limited and controlled way”

James Adam - Ruby Conf 2015

- reduce impact of monkey patching
- extend class locally
- available since Ruby 2.0
- works in class only
- activated in current and nested lexical scope

What about scope?

Lexical scope means that written code occures in the same code block:

```
class MySpace
  def songs
    ‘many’
  end

  def albums
    100
  end
end
```

same lexical scope

different lexical scope

```
class MySpace
  def songs
    ‘not many’
  end
end
```

Reopening a class or module means Ruby creates a new lexical scope!

A child class of a parent class has also a new lexical scope!

means refinements “using” in the parent class won’t work in the child class!

Opening a new file will also create a new lexical scope!

Blocks have their own lexical scope!

Refinements are lexically scoped!

Code examples

Real life example

Other ideas

When Monkey Patching, then do it the better way. Put them in CoreExtensions:

```
require ‘json’
module CoreExtensions
  module Hash
    module Sizer
      def size
        self.length * 2
      end

      def to_json
        JSON.generate(self)
      end
    end
  end
end
```

Your ideas?

- Helper with include
- Visitor pattern (https://en.wikipedia.org/wiki/Visitor_pattern)
- Dynamic instance methods (http://rohitrox.github.io/2013/07/02/ruby-dynamic-methods/)
- ???

A word of Warning

- some say it’s better than Monkey Patches but it is still the same in a way
- it needs to be examined further
- where are good examples how to use Refinements?
- why does nearly nobody use them?

Conclusion

Further reading and watching

RubyConf 2015 - Why is nobody using Refinements? by James Adam https://www.youtube.com/watch?v=qXC9Gk4dCEw

3 Ways to Monkey-patch Without Making a Mess by Justin Weiss
http://www.justinweiss.com/articles/3-ways-to-monkey-patch-without-making-a-mess/

Understanding Ruby Refinements and Lexical Scope by Starr Horne
http://blog.honeybadger.io/understanding-ruby-refinements-and-lexical-scope/

RubyDoc - Refinements
http://ruby-doc.org/core-2.3.0/doc/syntax/refinements_rdoc.html

Thank’s