ruby

Eine kurze Einführung

Warum nicht Java ?

http://rubyonrails.org/

Überblick

- ruby als Programmiersprache
- Alternativen
  - python, groovy
- rails
  - Entwicklung von Web-Anwendungen
  - Alternativen: grails, django
- Projekt

ruby

- 1995 entstanden
  - japanischer Nachfolger von perl
  - erst nach 2000 in Europa/USA bekannt
- OO-Sprache, Multi-Paradigmen
- dynamische Typisierung
- plattformunabhängig

dynamische, strenge Typisierung

```
>> i = "Hallo" # Ein String
=> "Hallo"
>> i.class
=> String

>> i = 1       # Eine Zahl
=> 1

>> puts "Hallo" + i
TypeError: can't convert Fixnum into String
        from (irb):348:in `+'
        from (irb):348
        from :0

>> i.class     # ist ein Objekt
=> Fixnum

>> i.methods   # mit Methoden
=> ["%", "odd?", "inspect", "prec_i", "<<", "tap", "div", "&", "clone",
">>", "public_methods", "object_id", "__send__",
"instance_variable_defined?", "equal?", "freeze", "to_sym", "*", "ord",
"+", "extend", "next", "send", "round", "pretty_print", "methods",
"pretty_print_inspect", "prec_f", "-", "even?", "singleton_method_added",
"divmod", "hash", "/", "integer?", "downto", "dup", "to_enum",
"instance_variables", "ri", "|", "eql?", "size", "id", "instance_eval",
usw.
```

reopening classes (Monkey patching)

```
?> class Fixnum
>>   def mal_zehn
>>     self*10
>>   end
>> end
=> nil

>> i = 5
=> 5
>> i.class
=> Fixnum
>> i.mal_zehn
=> 50

>> i.methods
=> ["%", "odd?", "inspect", "prec_i", "<<", "tap", "div", "&", "clone",
">>", "public_methods", "object_id", "__send__",
"instance_variable_defined?", "equal?", "freeze", "to_sym", "*", "ord",
"+", "extend", "next", "send", "round", "pretty_print", "methods",
"pretty_print_inspect", "prec_f", "-", "even?", "singleton_method_added",
"divmod", "hash", "/", "integer?", "downto", "dup", "to_enum",
"instance_variables", "ri", "|", "eql?", "size", "id", "instance_eval",
"truncate", "~", "to_i", "singleton_methods", "mal_zehn", "pretty_inspect"
```

def leitet eine Methodendefinition ein, letzter Ausdruck ist Rückgabewert

Monkey patching ist cool

```
>> class Fixnum
>>   def fib
>>     if self.zero?
>>       0
>>     elsif self == 1
>>       1
>>     else
?>       (self - 2).fib + (self - 1).fib
>>     end
>>   end
>> end

>> 2.fib
=> 1
>> (1...15).collect {|x| x.fib}
=> [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]

# Nur für Erwachsene
>> class String
>>   def length
>>     42
>>   end
>> end

>> "Hallo".length
=> 42
```

Objekte ohne Klassen

```
>> p = Object.new          # Eine Person
=> #<Object:0x1010e4ca8>

>> def p.SagWas
>>   puts "Hallo"
>> end
=> nil

>> p.class
=> Object

>> p.SagWas
Hallo
=> nil

# Wie ruft man unbekannte Methoden von Objekten ohne Klasse auf ?
>> p.send(:SagWas)
Hallo
=> nil
```

Es gibt auch Objekte ohne (explizite) Klasse: Gehören zur Klasse Object.

Objekte können dynamisch erzeugt werden.

Duck typing

```
>> s = Object.new         # Ein Student
=> #<Object:0x1010c8210>

>> def s.SagWas
>>   puts "Ein Student"
>> end

>> def s.TuWas
>> puts "Bier trinken"
>> end

>> s.TuWas
Bier trinken

>> s.SagWas
Ein Student

>> [s,p].each {|o|
?>   o.SagWas
>> }
Ein Student
Hallo
```

"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck." (James Whitcomb Riley)

Nicht die Klasse, sondern die Fähigkeiten (Methoden, genauer: die akzeptierten messages) definieren den Typ.

Klassen

```
>> class Student
>> attr_accessor :Name, :Matrikelnummer
>> end
=> nil

>> s = Student.new
=> #<Student:0x100551a80>

>> s.Name="Hans Mueller"
=> "Hans Mueller"
>> s.Matrikelnummer = 4711
=> 4711
>>
>> s
=> #<Student:0x100551a80 @Matrikelnummer=4711, @Name="Hans Mueller">
# Wie funktioniert das ?

>> s.instance_variables
=> ["@Matrikelnummer", "@Name"]
```

class leitet eine Klassendefinition ein
Methode initialize ist Konstruktor (Student.new ...)
Attribute müssen nicht explizit definiert werden, attr_accessor, attr_reader, attr_writer liefern Zugriff

Metaprogrammierung, Mixins

```
>>  module MyIntro
>>    def show_me
>>      self.instance_variables.each {|attr|
?>        puts "#{attr} is #{self.instance_variable_get attr}"
>>      }
>>    end
>>  end

>> s.extend MyIntro
=> #<Student:0x100551a80 @Matrikelnummer=4711, @Name="Hans Mueller">

>> s.show_me
@Matrikelnummer is 4711
@Name is Hans Mueller
=> ["@Matrikelnummer", "@Name"]
>>
```

Mixins erlauben die dynamische Erweiterung von Klassen durch module.

--> Rubys Alternative zur Mehrfachvererbung

Introspection

```
>> module Bean
>>   def method_missing(m, *args)   # wird aufgerufen, falls unbek. Methode
>>     if (/^get/.match(m.to_s))    # get am Anfang des Namens
>>      self.instance_variable_get "@#{m.to_s.gsub(/get/,'')}"
>>     else
?>        if (/^set/.match(m.to_s)) # set am Anfang des Namens
>>          self.instance_variable_set "@#{m.to_s.gsub(/set/,'')}", args[0]
>>        end
>>     end
>>   end
>> end
=> nil
>> => nil

>>  s.extend Bean
=> #<Student:0x10101d9f0 @Matrikelnummer=4711, @Name="Hans Mueller">

>> s.getName
=> "Hans Mueller"

>> s.setName "Hasi Meier"
=> "Hasi Meier"

>> s
=> #<Student:0x10101d9f0 @Matrikelnummer=4711, @Name="Hasi Meier">
>>
```


STOP

Variablen

```
>> i = 1              # Werden durch Zuweisung erzeugt
=> 1

>> C = 5              # Konstanten beginnen mit Grossbuchstaben
=> 5
>> C = 3
(irb):365: warning: already initialized constant C
=> 3

# Variablennamen können Sonderzeichen enthalten, CamelCase unüblich
>> ein_langer_name_ist_schoen = 1
=> 1

>> s = "Hallo"       # Ein String
=> "Hallo"
>> s.upcase          # Eine Methode der Klasse String, () nicht nötig
=> "HALLO"
>> t = 'Welt'        # Noch ein String
=> "Welt"
>> s + t             # Verkettung von Strings
=> "HalloWelt"
```

Name beginnt mit Kleinbuchstabe --> Variable
Name beginnt mit Grossbuchstabe --> Konstante
Name beginnt mit @ --> Instanzvariable
Name beginnt mit @@ --> Klassenvariable
Name beginnt mit $ --> globale Variable

Verzweigungen

```
>> i = 7                # Eine (ungerade) Zahl
=> 7

>> if (i % 2 == 0)
>>   puts "grade"
>> else
?>   puts "ungrade"
>> end
ungrade

# Oft:
>> puts "ungrade" unless (i % 2 == 0)
ungrade

>> puts "ungrade" if (i % 2 != 0)
ungrade

# z.B. Kommentare ignorieren
>> while line = gets
>>   next if /^#/.match line    # reguläre Ausdrücke in //, Zeilenanfang
>>   puts line
>> end
Hallo
Hallo
# Kommentar
Hallo
Hallo
```

Funktionen und Parameter

```
>> def tuwas(i)    # Eine Funktion mit Parameter i
>>   i+1
>> end

>> tuwas 3
=> 4
>> i=5
>> tuwas(i)
=> 6
>> i
=> 5

# aber :
>> def was(s)
>>   s.gsub!(/a/,'x')  # Ersetze a durch x
>> end
>> tst = "Hallo"

>> was(tst)
=> "Hxllo"
>> tst
=> "Hxllo"
```

Parameter sind Referenzen auf Objekte, werden immer by value übergeben.
Zuweisung erzeugt neues Objekt (siehe oben). Funktionen mit ! am Ende manipulieren den Inhalt. (Funktionen mit ? am Ende liefern true oder false zurück --> Konventionen)

Arrays

```
>> a = [1,2,"Hallo"]
=> [1, 2, "Hallo"]
>> a.length
=> 3
>> a[0]
=> 1
>> a[2]
=> "Hallo"
>> a[3]
=> nil
>> a[0].class
=> Fixnum
>> a[3].class
=> NilClass
>> a[2].class
=> String
>> a.last
=> "Hallo"
>> a.push "Welt"
=> [1, 2, "Hallo", "Welt"]
>> a.pop
=> "Welt"
>> a
=> [1, 2, "Hallo"]
```

Iterationen

```
5.times do
  puts "Hallo"
end


(1..3).each {|i|     # Ein Range-Objekt
  puts i
}

a.each do |x| # Block zwischen do und end wird für jedes Element ausgeführt
  puts x
end
1
2
Hallo

# {...} statt do ... end äquivalent
a.each {|x|
  puts x
}

["hi", "there"].collect { |word| word.capitalize }
=> ["Hi", "There"]
>>
```

Blöcke (closures)

```
 (1..3).each {|i|     # Ein Range-Objekt, each ist Methode, was ist {...}
  puts i
}
```

„Blocks are like alien logic“ [http://allaboutruby.wordpress.com/2006/01/20/ruby-blocks-101/]
Code als (implizite) Funktionsparameter !

```
>> a = [1,2]
=> [1, 2]
>> a.find {|x| x == 2}
=> 2
```

Erlauben die (einfache) Implementierung eigener Iteratoren

Dictionaries

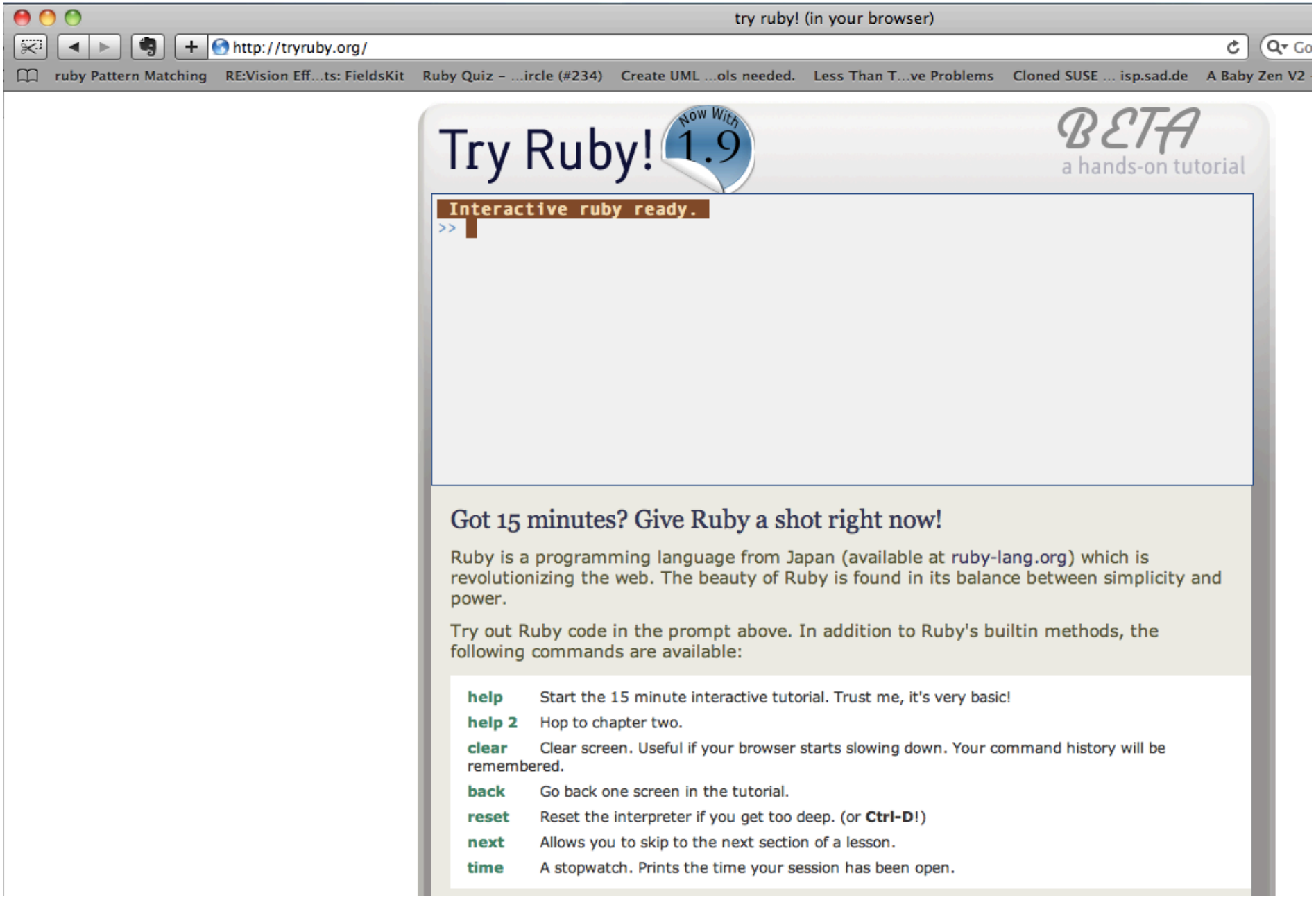
```
 >> Einwohner = Hash.new
=> {}
>> Einwohner["Ulm"]=122087
=> 122087
>> Einwohner["Dresden"]=521602
=> 521602
>> Einwohner["Heidenheim"]=50963
=> 50963
>> Einwohner
=> {"Ulm"=>122087, "Heidenheim"=>50963, "Dresden"=>521602}

>> Einwohner["Ulm"]
=> 122087

Einwohner.keys
=> ["Ulm", "Heidenheim", "Dresden"]
>> Einwohner.values
=> [122087, 50963, 521602]
>>

Einwohner.each {|key,value|
  puts "#{key} hat #{value} Einwohner"
}

Ulm hat 122087 Einwohner
Heidenheim hat 50963 Einwohner
Dresden hat 521602 Einwohner
```



http://tryruby.org/

STOP

Warum ist ruby (für uns) interessanter als andere Sprachen ??

rails

Framework Usage Statistics

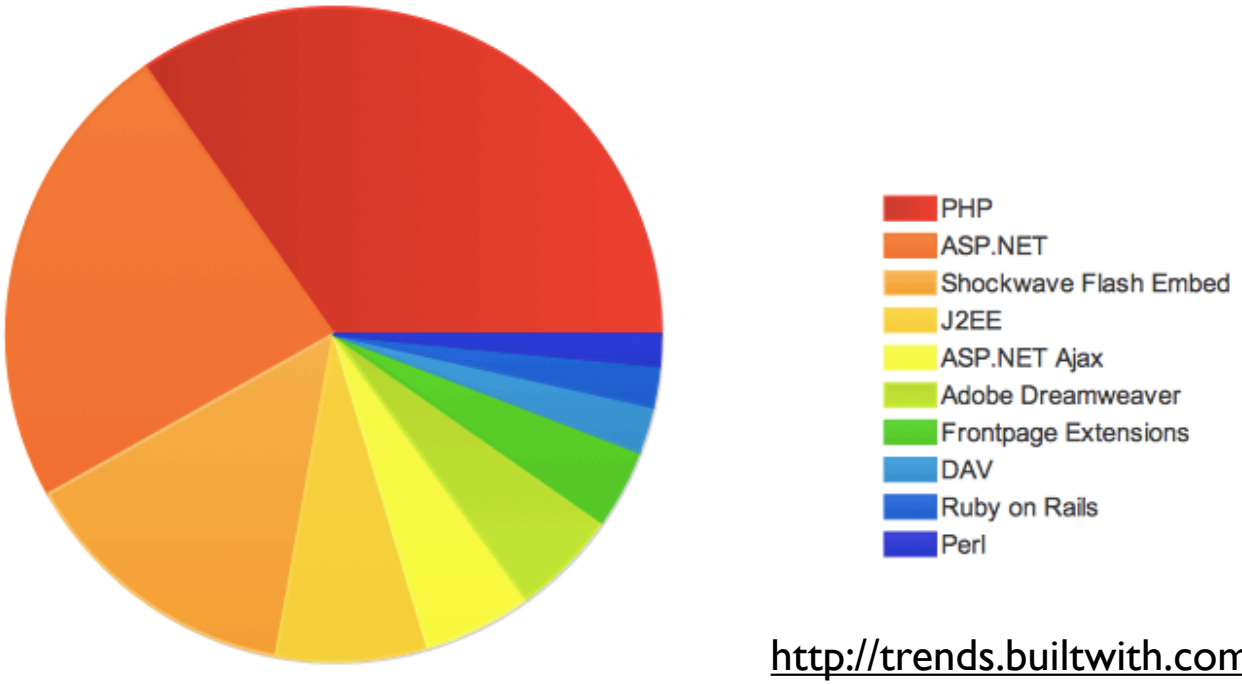
Overview of statistics for Framework technologies

Home | Frameworks

Home > Frameworks

Frameworks Distribution

Distribution is calculated from the top 10,000 websites on the internet. See the FAQ for more information. Last calculated on March 03 2011.



http://trends.builtwith.com/framework

Ruby on Rails Usage Statistics

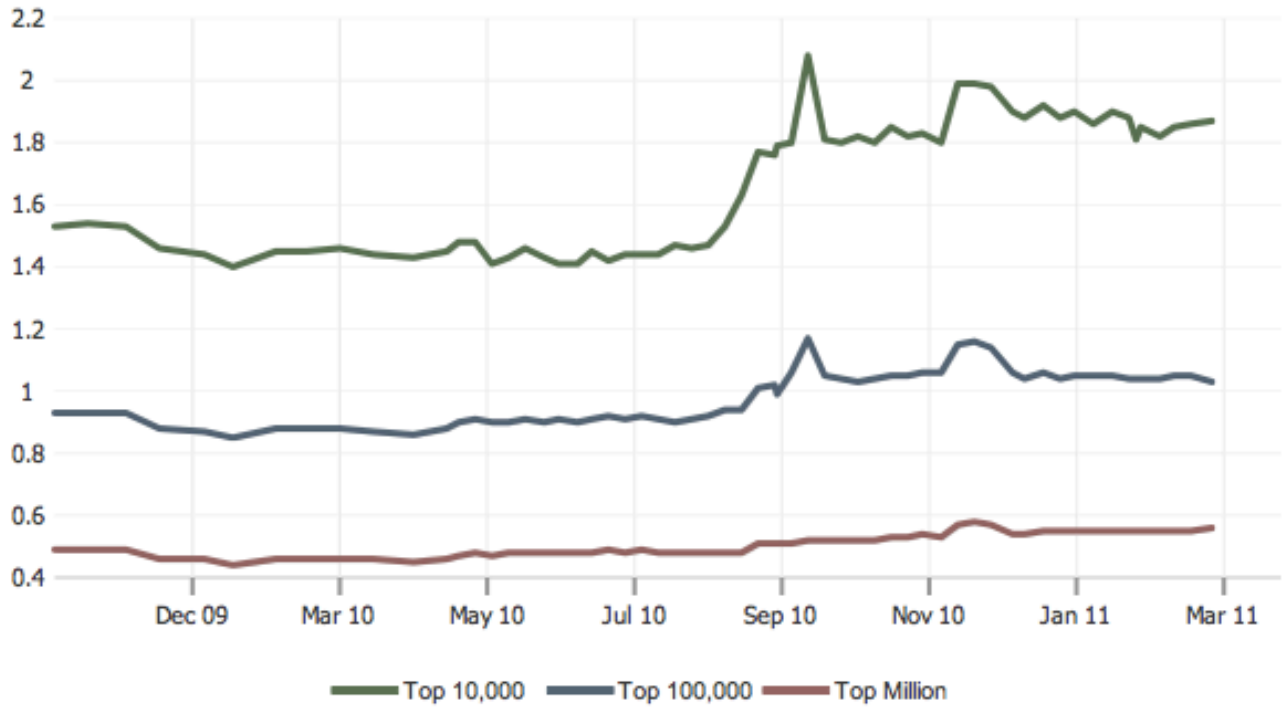
Websites using Ruby on Rails

Home | Frameworks | Ruby on Rails | Ruby on Rails Websites

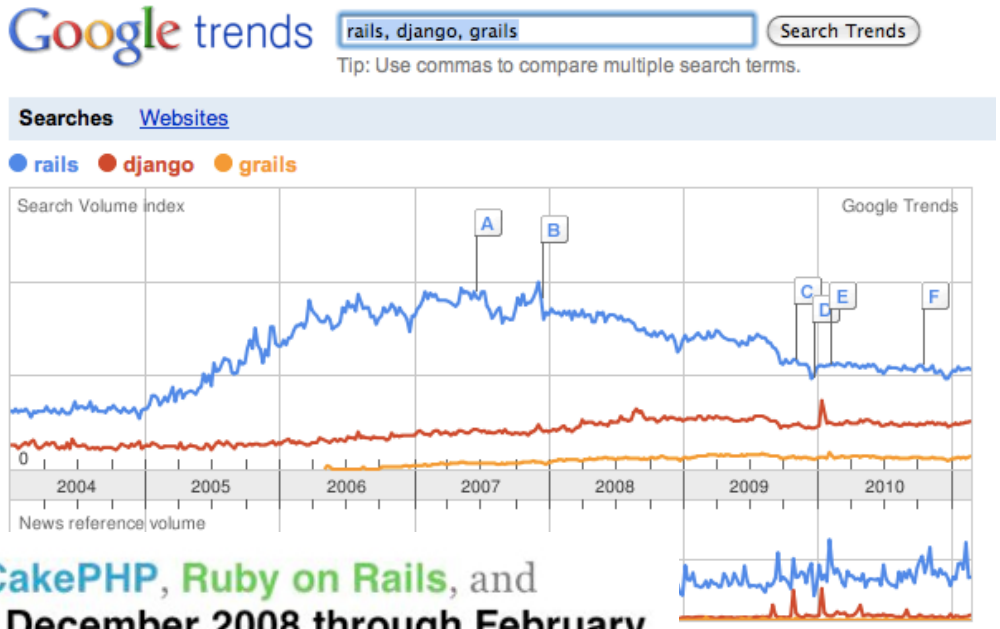
Home > Frameworks > Ruby on Rails Usage Statistics

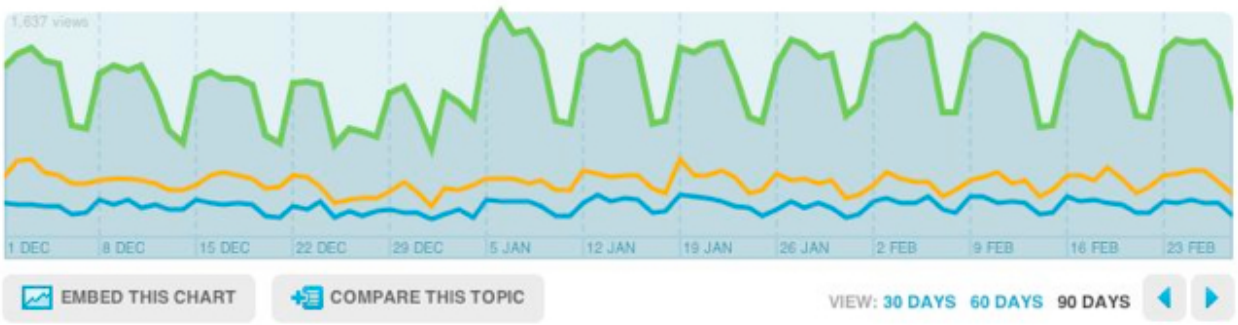
Ruby on Rails Usage Trends

Ruby on Rails is an open-source web framework that is optimized for programmer happiness and sustainable productivity.



Die Konkurrenz





THE JONAH GROUP

What they're saying...

"Rails is the most well thought-out web development framework I've ever used. And that's in a decade of doing web applications for a living. I've built my own frameworks, helped develop the Servlet API, and have created more than a few web servers from scratch. Nobody has done it like this before."

-James Duncan Davidson, Creator of Tomcat and Ant

"It is impossible not to notice Ruby on Rails. It has had a huge effect both in and outside the Ruby community... Rails has become a standard to which even well-established tools are comparing themselves to."

-Martin Fowler, Author of Refactoring, PoEAA, XP Explained

"After researching the market, Ruby on Rails stood out as the best choice. We have been very happy with that decision. We will continue building on Rails and consider it a key business advantage."

-Evan Williams, Creator of Blogger and ODEO

"Ruby on Rails is a breakthrough in lowering the barriers of entry to programming. Powerful web applications that formerly might have taken weeks or months to develop can be produced in a matter of days."

-Tim O'Reilly, Founder of O'Reilly Media

zurück zu ruby: Aufgaben

- HelloWorld
  - gibt den String Hello World aus
- Dreieck
  - gibt ein Dreieck aus Sternchen aus
- Primzahlen von 1 bis 1000 mit Sieb
- echo
- wc
- daytime server (TCP Port 13)
- Einfacher Web-Server (nur statische Seiten)