



Red Hot Ruby

前田 修吾

Ruby Association LLC

2009-04-09



自己紹介

- 所属
- Rubyとの出会い
- Rubyとの関わり

○●●所属

● Rubyアソシエーション

○ 副理事長

● ネットワーク応用通信研究所(NaCl)

○ 取締役

○●● Rubyとの出会い

- 1997年
- JavaHouse ML

○●● Rubyとの関わり

- 仕様提案・実装
- ライブラリの開発
- アプリケーションの開発
- サーバ管理



●●●本日のメニュー

● Rubyとは

● Rubyの歴史

● 流行の理由

● 言語の特徴

● 開発の動向

● 国際標準化



○●● Rubyとは

- オブジェクト指向言語
- スクリプト言語
- オープンソース
- Rubyの作者
- Rubyのコンセプト



○●● オブジェクト指向言語

- 本格的なオブジェクト指向言語
 - Smalltalkの影響
- すべてのデータはオブジェクト



○●● スクリプト言語

- 手軽なスクリプト言語
 - Perlの影響
- クラスを定義しなくてもいい



○●●オープンソース

- 誰でも自由に利用できる
- デュアルライセンス
 - GPL
 - 独自ライセンス



○●● Rubyの作者

- まつもとゆきひろさん
- Rubyアソシエーション理事長
- ネットワーク応用通信研究所フェ
ロー



○●● Rubyのコンセプト

● プログラマに最適化

○ 「コンピュータに」ではない

○ プログラマ = まつもとさん

○ プログラマ != 初心者

● プログラムを簡潔に

○ 「言語仕様を簡潔に」ではない



○●● Rubyの歴史

1993-02-24 誕生
1995-12-21 fjで公開
1996-12-25 ruby 1.0
2000-11-29 Perl/Ruby Conference
2001-10-12 Ruby Conference
2005-12-14 Ruby on Rails 1.0
2006-06-10 日本Rubyカンファレンス(RubyKaigi)
2009-01-31 Ruby 1.9.1



誕生のきっかけ

- オブジェクト指向の書籍
 - 言語を作りながら学ぶ
- 書籍自体は出版されず



○●● Rubyの誕生日

● 1993年2月24日

○ 名前が決まった日

● Javaより古い

○ OakからJavaに変わったのは1995年



● ● ● 名前の由来

```
keiju> そうそう．言語名考えた？  
matz> 日立の子会社が広告を打ってた  
matz> うーん、shell に十分似ているなら Tish.  
matz> でももっと格好良い名前が欲しいなあ  
  
...  
keiju> ruby  
keiju> やはり宝石名でない？  
matz> 「ルビを振る」のルビ？  
matz> 何で宝石名なんだ  
matz> 三菱の影響か？  
keiju> perl  
matz> なるほど
```



後付けの理由

- Perlの次の言語

- 誕生石

- pearlは6月、rubyは7月

- 活字の大きさ

- pearlは5pt、rubyは5.5pt



●●● 流行の理由

- Ruby on Railsでの採用
 - Web開発での利用が急増
- 趣味の言語からビジネスの言語へ



○●● Ruby on Railsとは

- Webアプリケーションフレームワーク
- David Heinemeier Hanssonさん作



なぜ採用されたか?

もしWebアプリケーションの開発を今後も続けるのであれば、我慢して使うものじゃなくて、心から愛せるツールを使うべきだって思い立ったんです。

...(略)...

一日経つと「Rubyが本当に好き」になり、一週間経つと「PHPには戻れない」状況になりました。Rubyの熟練度がPHPでのそれを上回るには、一ヶ月もかかりませんでした。Rubyは、それはもう、ものすごくフィットしたんです。私の脳に完璧にフィットしました。それからは楽しく、より良く作業が行えるようになりました。

Ruby on Rails: David Heinemeier Hanssonへのインタビュー
<http://capsctrl.que.jp/kdmsnr/wiki/transl/?AnInterviewWithDHH>



言語の特徴

- スクリプト言語
- オブジェクト指向言語
- 動的言語
- 関数型言語?



○●● スクリプト言語

- インタープリタ
- 簡潔な記法
- テキスト処理機能



○●● インタープリタ

```
$ ruby hello.rb  
$ ruby -e 'puts "hello world" '  
$ ruby -pe 'gsub(/perl/, "ruby")'
```



簡潔な記法

```
class Hello
  def say(whom = "world")
    puts "hello " + whom
  end
end
hello = Hello.new
hello.say("shugo") #=> hello shugo
hello.say          #=> hello world
```



●●● テキスト処理機能

```
s = "perl is cool".sub(/perl/, "ruby")
s = "hello world\n".chop
s = "  hello wolrd  ".strip
words = "ruby perl".split
word = "ruby perl".slice(/\w+/)
name = "def foo".slice(/def (\w+)/, 1)
```



○●●オブジェクト指向言語

- 純粋
- クラスベース
- Mix-in
- オブジェクトベース





純粹

● すべてのデータがオブジェクト

○ 数値

○ 文字列

○ 配列



○●● クラスベース

- オブジェクトはクラスに属する
- クラスによってオブジェクトの振舞
が決まる
- 単一継承



○●● Mix-in

● 限定された多重継承

○ 複数のクラスは継承できない

○ モジュールなら複数継承できる

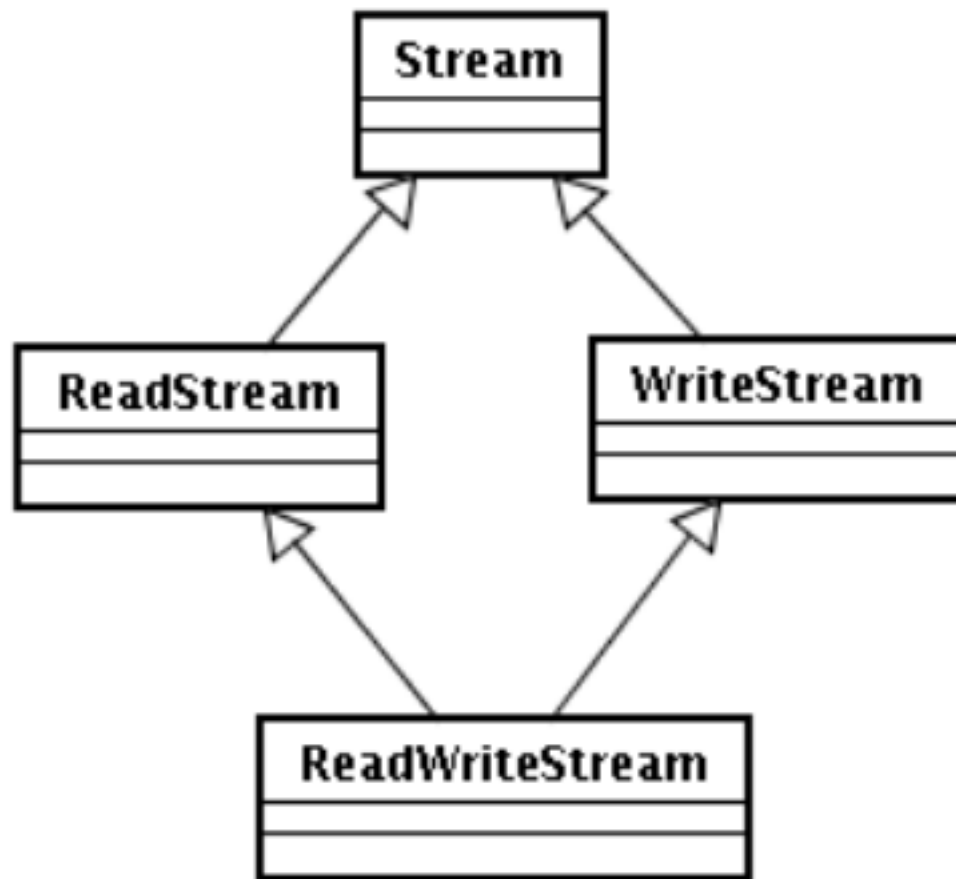
● モジュールとは

○ クラスと同じようなもの

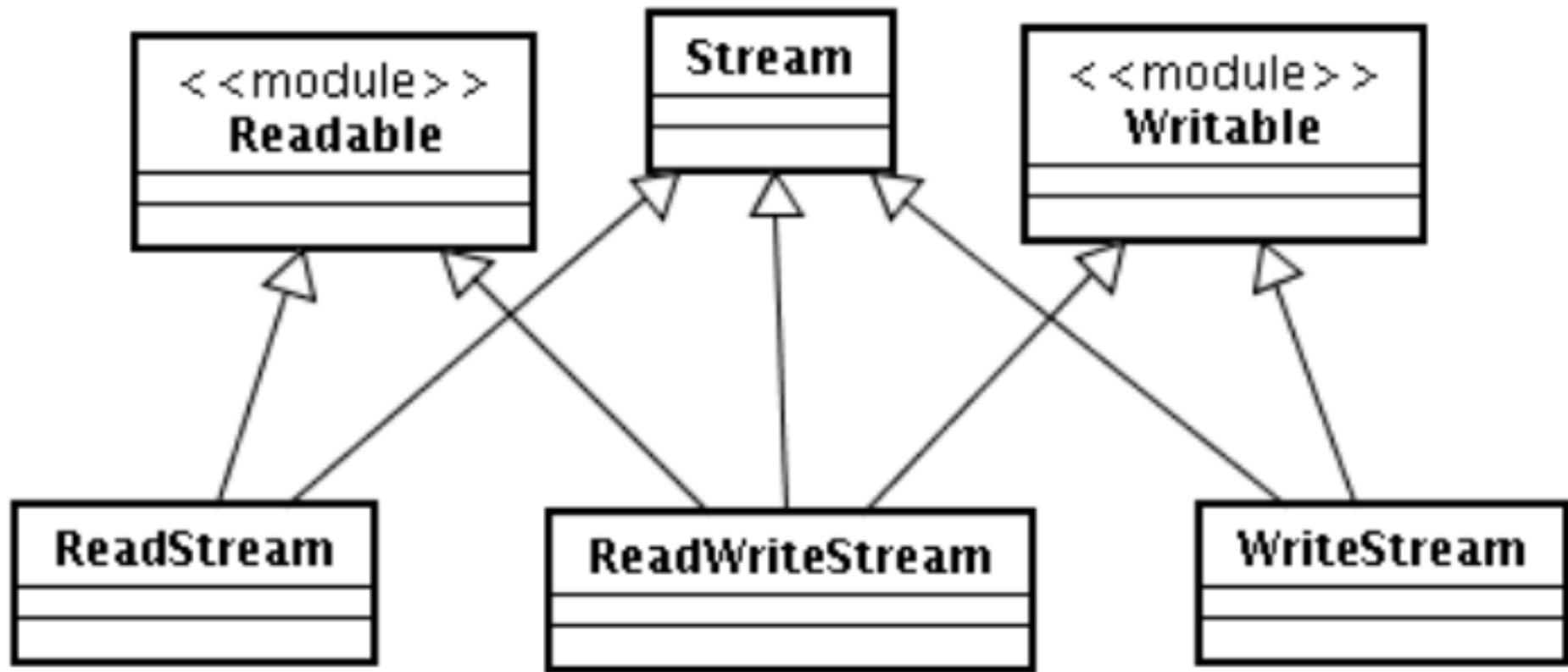
○ インスタンス化できない



多重継承の例



●●● Mix-inの例



○●●オブジェクトベース

- クラス定義は必須ではない

```
duck = Object.new
def duck.quack
  puts "クワックワッ"
end
duck.quack
```



動的言語

● 静的型はない

○ 変数に型は指定しない

○ コンパイル時に型情報は得られない

● ほとんどのことを実行時に行う

○ クラス定義

○ メソッド定義



○●● Duck Typing

- 鴨のように歩き、鴨のように鳴くものは、鴨に違いない

```
class Duck; def walk; end end
class Fox; def walk; end end
def foo(duck)
  duck.walk
end
foo(Duck.new)
foo(Fox.new)
```



関数型言語?

- ブロックをメソッドに渡せる

```
p ["1", "2", "3"].collect { |s|  
  s.to_i  
} #=> [1, 2, 3]
```



関数型言語?(2)

- オブジェクト化することも可能

```
plus = lambda { |x, y|  
  x + y  
}  
p plus.call(2, 3) #=> 5
```



関数型言語?(3)

- Ruby 1.9ではこんな書き方も

```
Y = ->(f) {  
  ->(x) {  
    f[->(arg) { x[x][arg] }]  
  }[  
    ->(x) {  
      f[->(arg) { x[x][arg] }]  
    }  
  ]  
}
```



○●● 開発の動向

- バージョン体系
- 開発ブランチ
- 各バージョンの位置付け
- その他の処理系



バージョン体系

バージョンの付け方:

<MAJOR> . <MINOR> . <TEENY> -p<PATCHLEVEL>

例: 1.8.6-p368

MAJOR = 1

MINOR = 8

TEENY = 6

PATCHLEVEL = 368



バージョン体系(2)

MAJOR	大きな変更
MINOR	互換性のない変更も許容
TEENY	互換性を確保した変更のみ
PATCHLEVEL	バグフィックスのみ



開発ブランチ

- trunk(1.9)

- 1.9開発用(幹)

- ruby_1_8

- 1.8開発用・非互換な修正はNG

- ruby_1_8_x, ruby_1_9_x

- 保守ブランチ・バグ修正のみ



○●●各バージョンの位置付け

● 1.8系

● 1.9系



○●● 1.8系

- 安定版
 - 業務での利用におすすめ
- ちょっと遅い



1.8系の現在の開発方針

- 1.9系で導入された一部の文法・ライブラリをサポート
- 1.9系への移行をスムーズに
- 一部の反対意見
- 1.8.5より前を知らないせい?



○●● 1.9系

● 次期安定版

○ そろそろ業務での利用を検討できるレベル

● 他言語化

● 高速化



○●● その他の処理系

- JRuby
- Rubinius
- IronRuby
- MacRuby
- MagLev



●●● 国際標準化

- IPA公募事業
- 背景
- 開発への配慮



●●● IPA公募事業

- Rubyの国際標準化に関する調査
 - 標準仕様の草案作成が主な内容
 - NaClが受託



●●●背景

- 政府調達
- 安定した仕様へのニーズ



政府調達

- 政府調達の基本指針
 - 総務省
- オープンな標準
 - 特定製品の名指しを避ける



安定した仕様へのニーズ

- 処理系の開発
- アプリケーション開発
- 人材育成
 - テキスト
 - 試験問題



開発への配慮

● バージョン間の互換性

○ 1.8/1.9の双方が合致する仕様

● 処理系の互換性

○ 複数の処理系が合致する仕様

○ MRI, JRuby, Rubinius, IronRuby

進捗状況

- 文法/意味の大部分を記述
- 標準化検討WGでのレビュー・議論
- 処理系開発者の一部によるレビュー
- まだ一般に公開できるレベルではない



○●●まとめ

- Rubyはプログラマのための言語
- Railsによって、趣味の言語から業務で使う言語へ
- 活発な開発と標準化によって、流行からメインストリームへ