DRb

RMI For Smart People

Brian Sletten
brian@bosatsu.net

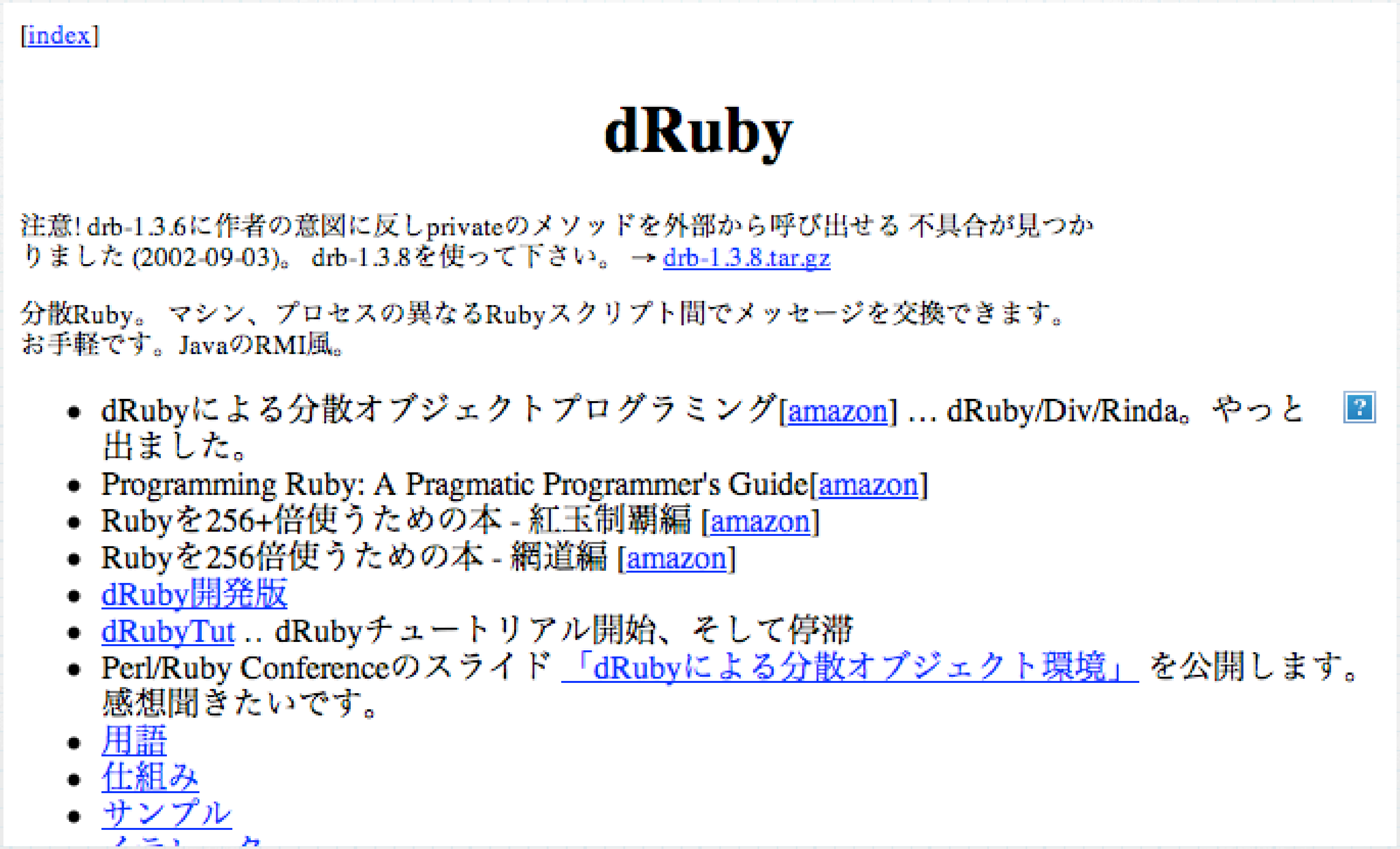
David Sletten
david@bosatsu.net

菩薩相談

DRb

* Distributed Ruby
* Written by Masatoshi Seki
* Ruby Objects in one process can easily call Ruby Objects in another process
* Ruby's answer to RMI
* It's soooooooooo easy...

Easy-Peasy Japanesey



[index]

dRuby

注意! drb-1.3.6に作者の意図に反しprivateのメソッドを外部から呼び出せる 不具合が見つかりました (2002-09-03)。 drb-1.3.8を使って下さい。 → drb-1.3.8.tar.gz

分散Ruby。 マシン、プロセスの異なるRubyスクリプト間でメッセージを交換できます。
お手軽です。JavaのRMI風。

- dRubyによる分散オブジェクトプログラミング[amazon] … dRuby/Div/Rinda。やっと出ました。
- Programming Ruby: A Pragmatic Programmer's Guide[amazon]
- Rubyを256+倍使うための本 - 紅玉制覇編 [amazon]
- Rubyを256倍使うための本 - 網道編 [amazon]
- dRuby開発版
- dRubyTut .. dRubyチュートリアル開始、そして停滞
- Perl/Ruby Conferenceのスライド 「dRubyによる分散オブジェクト環境」 を公開します。感想聞きたいです。
- 用語
- 仕組み
- サンプル

DRb

* All Ruby
* Objects are used as if they were local
* Supports call by value and call by reference
* Supports exceptions, blocks and multithreaded use
* Not a ton of code

Not a Lot of Code

```
Maze:/usr/local/lib/ruby/1.8/drb brian$ wc `ls *.rb`
     144      307     2576 acl.rb
    1760     7206    52865 drb.rb
      16       32      253 eq.rb
      64      115     1081 extserv.rb
      96      183     1689 extservm.rb
     122      204     1925 gw.rb
      36       67      775 invokemethod.rb
      22       36      369 observer.rb
     190      477     5015 ssl.rb
      91      172     1507 timeridconv.rb
     108      259     2471 unix.rb
    2649     9058    70526 total
```

```
Maze:/usr/local/lib/ruby/1.8/drb brian$ cat *.rb | grep -v "^[ ]*#.*" | wc
    1896    3845   38677
```

DRb Server On 1 Slide

```ruby
require 'drb'

class ProprietaryServer
  def mask(cc_num)
    fields = cc_num.split("-")
    return "XXXX-XXXX-XXXX-#{fields[3]}"
  end

  def transform(cc_num)
    return cc_num.reverse
  end
end

server = ProprietaryServer.new
DRb.start_service("druby://localhost:9000", server)
DRb.thread.join
```

DRb Server On 1 Slide

```ruby
require 'drb'

class ProprietaryServer
  def mask(cc_num)
    fields = cc_num.split("-")
    return "XXXX-XXXX-XXXX-#{fields[3]}"
  end

  def transform(cc_num)
    return cc_num.reverse
  end
end

server = ProprietaryServer.new
DRb.start_service("druby://localhost:9000", server)
DRb.thread.join
```

DRb Server On 1 Slide

```ruby
require 'drb'

class ProprietaryServer
  def mask(cc_num)
    fields = cc_num.split("-")
    return "XXXX-XXXX-XXXX-#{fields[3]}"
  end

  def transform(cc_num)
    return cc_num.reverse
  end
end

server = ProprietaryServer.new
DRb.start_service("druby://localhost:9000", server)
DRb.thread.join
```

DRb Server On 1 Slide

```ruby
require 'drb'

class ProprietaryServer
  def mask(cc_num)
    fields = cc_num.split("-")
    return "XXXX-XXXX-XXXX-#{fields[3]}"
  end

  def transform(cc_num)
    return cc_num.reverse
  end
end

server = ProprietaryServer.new
DRb.start_service("druby://localhost:9000", server)
DRb.thread.join
```

DRb Server On 1 Slide

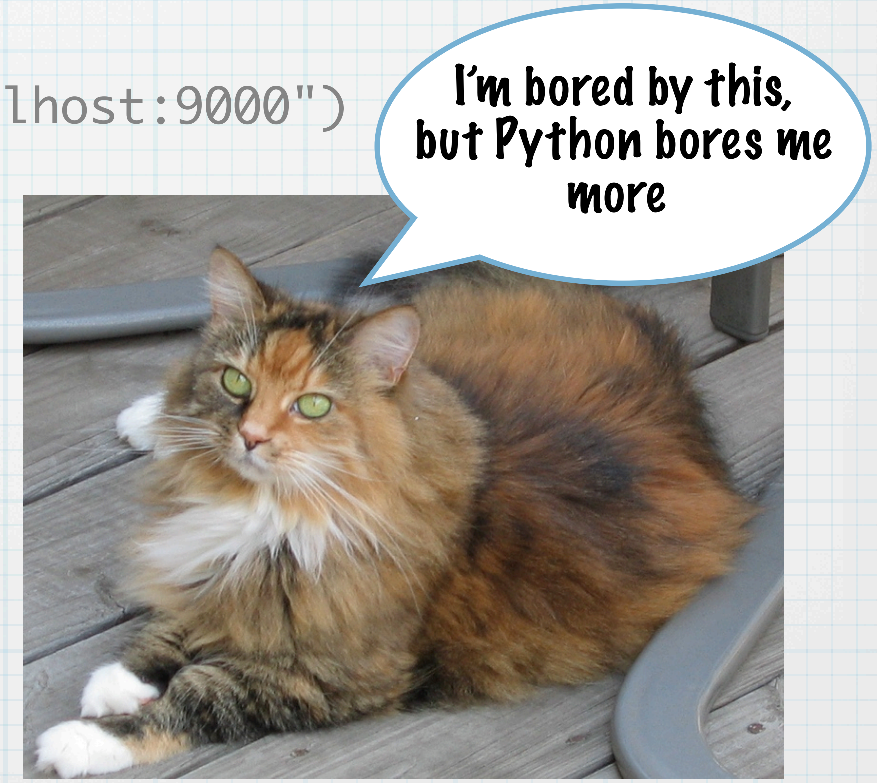
```
require 'drb'

class ProprietaryServer
  def mask(cc_num)
    fields = cc_num.split("-")
    return "XXXX-XXXX-XXXX-#{fields[3]}"
  end

  def transform(cc_num)
    return cc_num.reverse
  end
end

server = ProprietaryServer.new
DRb.start_service("druby://localhost:9000",
server)
DRb.thread.join
```

DRb Server On 1 Slide

```
require 'drb'

class ProprietaryServer
  def mask(cc_num)
    fields = cc_num.split("-")
    return "XXXX-XXXX-XXXX-#{fields[3]}"
  end

  def transform(cc_num)
    return cc_num.reverse
  end
end

server = ProprietaryServer.new
DRb.start_service("druby://localhost:9000", server)
DRb.thread.join
```

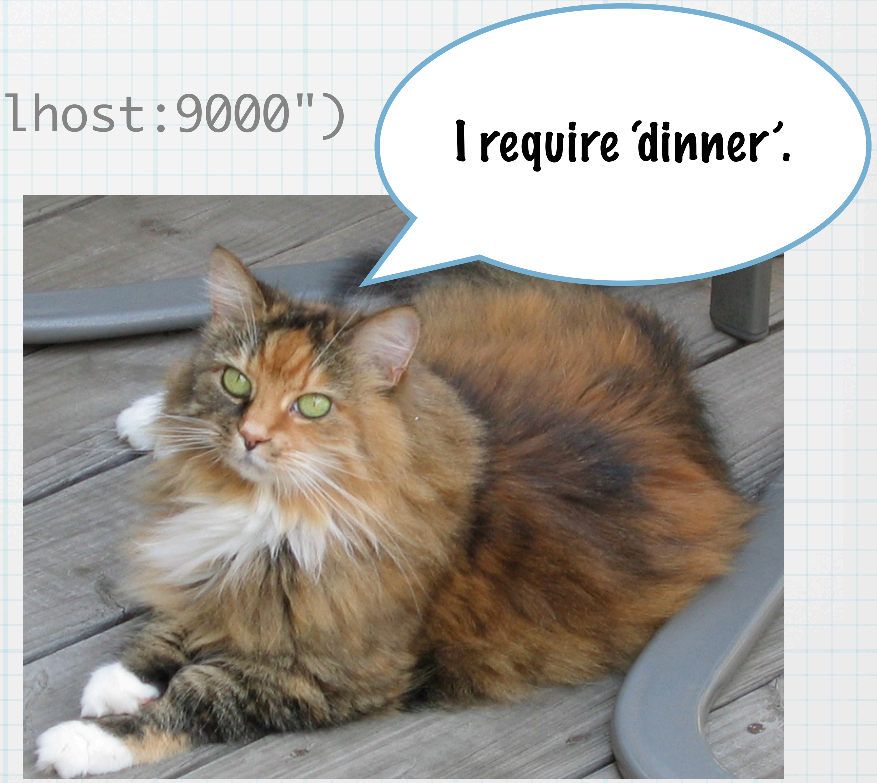
DRb Client On <1 Slide (plus my cat)

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")
cc_num = "5438-0166-8187-9942"

puts(obj.mask(cc_num))
puts(obj.transform(cc_num))
```



DRb Client On <1 Slide (plus my cat)

```ruby
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")
cc_num = "5438-0166-8187-9942"

puts(obj.mask(cc_num))
puts(obj.transform(cc_num))
```

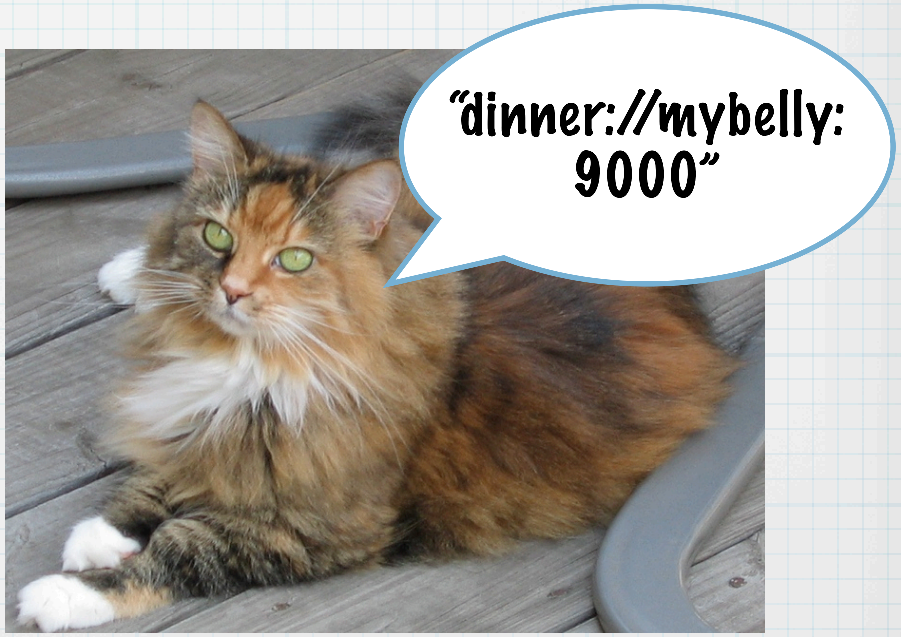


DRb Client On <1 Slide (plus my cat)

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")
cc_num = "5438-0166-8187-9942"

puts(obj.mask(cc_num))
puts(obj.transform(cc_num))
```

DRb Client On <1 Slide (plus my cat)

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")
cc_num = "5438-0166-8187-9942"

puts(obj.mask(cc_num))
puts(obj.transform(cc_num))
```



DRb Client On <1 Slide (plus my cat)

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")
cc_num = "5438-0166-8187-9942"

puts(obj.mask(cc_num))
puts(obj.transform(cc_num))
```

Don't bother with the credit card, it ain't real. I already tried it.

DRb Client On <1 Slide (plus my cat)

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")
cc_num = "5438-0166-8187-9942"

puts(obj.mask(cc_num))
puts(obj.transform(cc_num))
```

Another DRb Server

```
#
#    DRb server with access control.
#

require 'drb'
require 'drb/acl'

class IceCreamFlavorStore
  def initialize
    @favorites = {}
  end

  def get_favorite(person)
    @favorites[person]
  end

  def set_favorite(person, flavor)
    @favorites[person] = flavor
  end
end
```

```
#
#    Access control list
#
acl = ACL.new(%w[deny all
               allow localhost])


DRb.install_acl(acl)
DRb.start_service("druby://
localhost:9000",
    IceCreamFlavorStore.new)
DRb.thread.join
```

Another DRb Server

```
#
#    DRb server with access control.
#

require 'drb'
require 'drb/acl'

class IceCreamFlavorStore
  def initialize
    @favorites = {}
  end

  def get_favorite(person)
    @favorites[person]
  end

  def set_favorite(person, flavor)
    @favorites[person] = flavor
  end
end
```

```
#
#    Access control list
#
acl = ACL.new(%w[deny all
                 allow localhost])

DRb.install_acl(acl)
DRb.start_service("druby://localhost:
9000",
    IceCreamFlavorStore.new)
DRb.thread.join
```

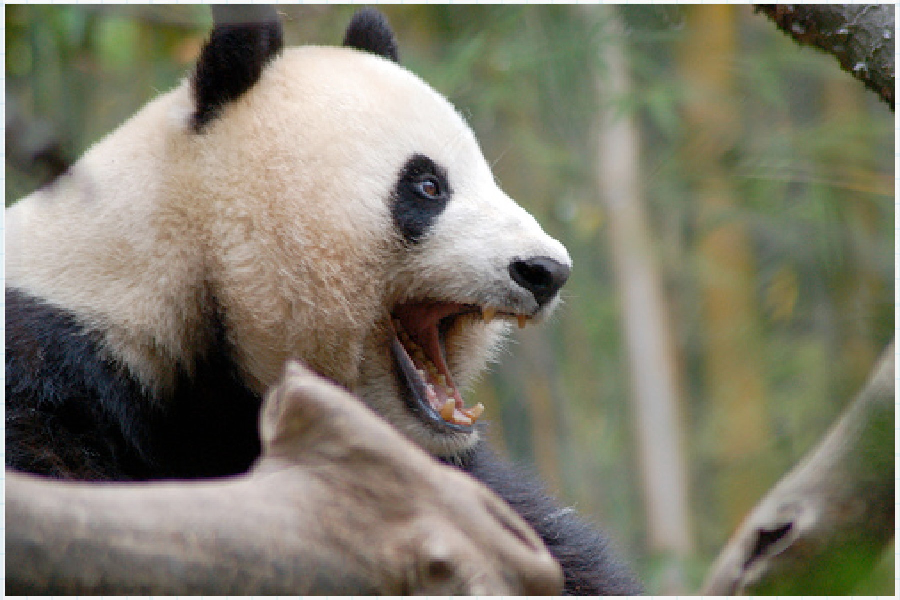
Another DRb Client

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")

obj.set_favorite("Bob", "Vanilla")
obj.set_favorite("Tina", "Mocha")
obj.set_favorite("Ferd", "Peach Nugget")
obj.set_favorite("Tallulah", "Coconut Cream")

puts("Bob's favorite flavor is #{obj.get_favorite("Bob")}")
puts("Tallulah's favorite flavor is #{obj.get_favorite("Tallulah")}")
```

YADC

```ruby
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")

puts("Tina's favorite flavor is #{obj.get_favorite("Tina")}")
puts("Ferd's favorite flavor is #{obj.get_favorite("Ferd")}")
```

Yowza!



```
ro = DRbObject::new_with_uri("druby://your.server.com:8989")
class << ro
    undef :instance_eval  # force call to be passed to remote object
end

ro.instance_eval("`rm -rf *`")
```

http://www.flickr.com/photos/amberlion/164176881/

Phew!

```
$SAFE = 1   # disable eval() and friends

DRb.start_service("druby://yourserver.com:8989", myObj )
DRb.thread.join
```

http://www.flickr.com/photos/amberlion/164176877

Marshaling

- DRb uses DRbMessage (although you don't care) to serialize objects
- Uses Marshal library

Marshal Write

```
a = [1, "two", 3.0]

#
#    Dump to file.
#
File.open("array.out", "w+") do |f|
  Marshal.dump(a, f)
end

#
#    Dump as String. First 2 bytes indicate version number. Normally, marshaled
#    data can only be restored by programs using same major version number and
#    greater than or equal minor version number of Ruby.
#
s = Marshal.dump(a)
puts("Minor version number: #{s[0]}")
puts("Major version number: #{s[1]}")
```

Marshal Read

```
a = []
File.open("array.out")
do |f|
  a = Marshal.load(f)
end

puts(a)
```

Custom Marshaling

```ruby
class Person
  attr_accessor :first_name, :last_name, :sex
  attr_reader :birthdate, :age

  def initialize(first_name, last_name, sex, birthdate)
    @first_name = first_name
    @last_name = last_name
    @sex = sex
    @birthdate = birthdate
    @age = compute_age
  end

  def marshal_dump
    [@first_name, @last_name, @sex, @birthdate]
  end

  def marshal_load(data)
    @first_name, @last_name, @sex, @birthdate = data
  end

  def to_s
    "Person name: #{@first_name} #{@last_name} sex: #{@sex} birthdate: #{@birthdate} age: #{@age.to_i}"
  end

  private

  def compute_age
    (Time.now - @birthdate) / (365 * 86400)
  end
end
```

Custom Marshal Write

```
require 'person'

p = Person.new("Barry", "Methylthwacker", "M", Time.local(1966, 5, 12))

File.open("person.out", "w+") do |f|
  Marshal.dump(p, f)
end
```

Custom Marshal Read

```
require 'person'

p = nil
File.open("person.out") do |f|
  p = Marshal.load(f)
end

puts(p)
```

Oops!!

```
Person name: Barry Methylthwacker sex: M birthdate: Thu May 12 00:00:00 EDT 1966 age: 0
```

$\nearrow$ Forgot to recompute the age!

Fixed Custom Marshaling

```ruby
def marshal_load(data)
  @first_name, @last_name, @sex, @birthdate = data
  @age = compute_age
end
```

Custom Marshal Server

```ruby
require 'drb'
require 'person1'

class Company
  def initialize
    @employees = {}
  end

  def add_employee(person)
    @employees[person.last_name] = person
  end

  def get_employee(name)
    @employees[name]
  end

  def get_employees
    @employees
  end
end

server = Company.new
DRb.start_service("druby://localhost:9000",
    server)
DRb.thread.join
```

Custom Marshal Client

```
require 'drb'
require 'person1'

DRb.start_service

company = DRbObject.new(nil, "druby://localhost:9000")

company.add_employee(Person.new("Karen", "Medcamp", "F", Time.local(1972, 4, 1)))
company.add_employee(Person.new("Sally", "Breene", "F", Time.local(1976, 7, 4)))
company.add_employee(Person.new("Tara", "Fuddle", "F", Time.local(1979, 2, 29)))

company.get_employees.values.each do |employee|
  puts(employee)
end
```

Using SSL w/ DRb

* Examples taken from segment7.net (see References)
* First, download QuickCert
  * http://segment7.net/projects/ruby/QuickCert/
* sudo make install
* This puts QuickCert in /usr/local/bin

Build Certs

* Create a QuickCert config file called qc_config
* Run QuickCert in dir w/ this file

```
full_hostname = `hostname`.strip
domainname = full_hostname.split('.')[1..-1].join('.')
hostname = full_hostname.split('.')[0]

CA[:hostname] = hostname
CA[:domainname] = domainname
CA[:CA_dir] = File.join Dir.pwd, "CA"
CA[:password] = '1234'

CERTS << {
  :type => 'server',
  :hostname => 'localhost',
  :password => '5678',
}

CERTS << {
  :type => 'client',
  :user => 'brian',
  :email => 'brian@bosatsu.net',
}
```

DRb SSL Server

```
require 'drb'
require 'drb/ssl'

here = "drbssl://localhost:3456"

class HelloWorld
  include DRbUndumped

  def hello(name)
    "Hello, #{name}."
  end
end

config = {
  :SSLPrivateKey =>
    OpenSSL::PKey::RSA.new(File.read("localhost/localhost_keypair.pem")),
  :SSLCertificate =>
    OpenSSL::X509::Certificate.new(File.read("localhost/cert_localhost.pem")),
}

DRb.start_service here, HelloWorld.new, config
DRb.thread.join
```

DRb SSL Client on < 1 Slide (plus Head Hurting)

```
require 'drb'
require 'drb/ssl'

there = "drbssl://localhost:3456"

config = {
  :SSLVerifyMode =>
OpenSSL::SSL::VERIFY_PEER,
  :SSLCACertificateFile => "CA/cacert.pem",
}

DRb.start_service nil, nil, config
h = DRbObject.new nil, there

while line = gets
  puts h.hello(line.chomp)
end
```

DRb SSL Server w/ Client Auth!

```
config = {
  :SSLVerifyMode => OpenSSL::SSL::VERIFY_PEER |
                           OpenSSL::SSL::VERIFY_FAIL_IF_NO_PEER_CERT,
  :SSLPrivateKey =>
    OpenSSL::PKey::RSA.new(File.read("localhost/localhost_keypair.pem")),
  :SSLCertificate =>
    OpenSSL::X509::Certificate.new(File.read("localhost/cert_localhost.pem")),
  :SSLCACertificateFile => "CA/cacert.pem"
}
```

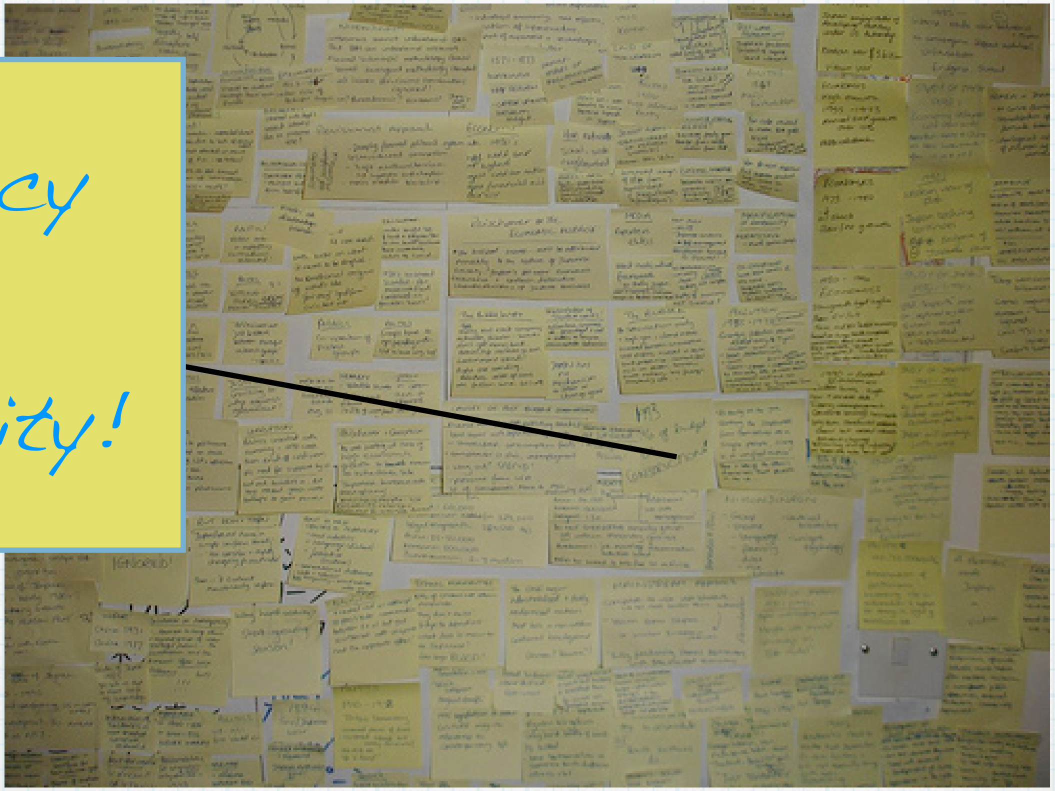
DRb SSL Client w/ Cert On < 1 Page (Plus Head Assplode)

```
require 'drb'
require 'drb/ssl'

there = "drbssl://localhost:3456"

config = {
  :SSLVerifyMode => OpenSSL::SSL::VERIFY_PEER,
  :SSLCACertificateFile => "CA/cacert.pem",
  :SSLPrivateKey =>
    OpenSSL::PKey::RSA.new(File.read("brian/
brian_keypair.pem")),
  :SSLCertificate =>
    OpenSSL::X509::Certificate.new(File.read("brian/
cert_brian.pem")),
}

DRb.start_service nil, nil, config
h = DRbObject.new nil, there

while line = gets
  puts h.hello(line.chomp)
end
```

Don't Forget

http://www.flickr.com/photos/tamegoeswild/146124793/

Don't Forget



Concurrency is your Responsibility!

http://www.flickr.com/photos/tamegoeswild/146124793/

Bad Server (no doughnut!)

```
require 'drb'

class ComputationServer
  def initialize
    @result = 0
  end

  def compute(x)
    @result = x
    f
    g
    return @result
  end

  def f
    sleep(rand(3))
    @result += 1
  end

  def g
    sleep(rand(3))
    @result *= 2
  end
end

server = ComputationServer.new
DRb.start_service("druby://localhost:9000",
    server)
DRb.thread.join
```

Don't Blame the Victim

```
require 'drb'

DRb.start_service

obj = DRbObject.new(nil, "druby://localhost:9000")

puts(obj.compute(5))
```

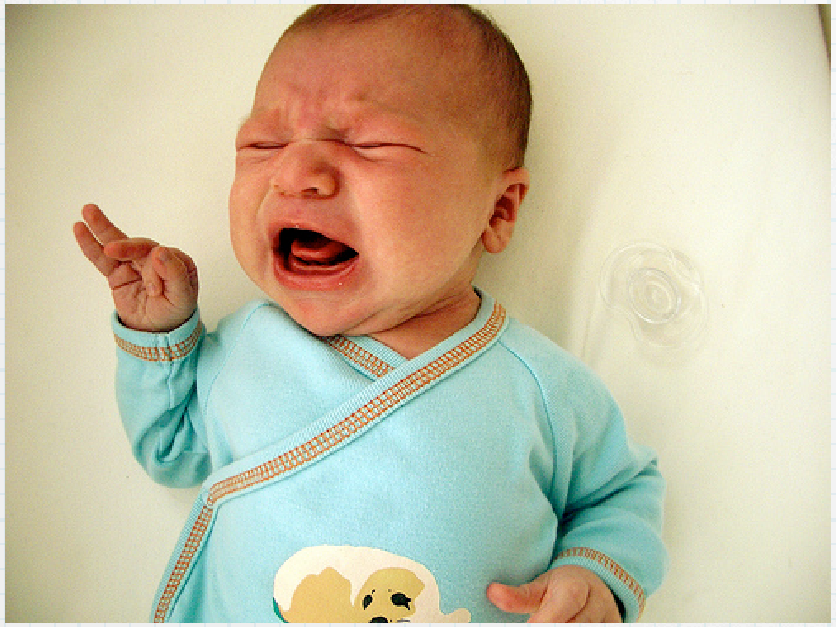
Fixed Server

```
def initialize
  @result = 0
  @mutex = Mutex.new
end

def compute(x)
  @mutex.synchronize do
    @result = x
    f
    g
    return @result
  end
end
```

Other Uses of DRb

* Background DB to offload long-running tasks from Rails
    * http://backgroundrb.rubyforge.org/
* Rinda Tuplespace implementation
    * http://stdlib.rubyonrails.org/libdoc/rinda/rdoc/index.html

Things About DRb That Might Make You Cry



* Ruby Only
* Security issues if you aren't careful
* Performance can be a problem if you are silly

http://www.flickr.com/photos/finnern/150664612/

In General, We Dig DRb

* Simple
* Lightweight
* Performance is ok
* Useful for Prototyping
* Useful for Debugging

http://www.flickr.com/photos/babytomtom/192997899/

References

RDoc	http://www.ruby-doc.org/stdlib/libdoc/drb/rdoc/index.html
Slides	http://www.bosatsu.net/talks/DRb.pdf
Examples	http://www.bosatsu.net/talks/examples/DRb.zip
DRb w/ SSL	http://segment7.net/projects/ruby/drb/DRbSSL/
"The Ruby Way"	http://tinyurl.com/p4uuc