V0.7

dRuby 20th anniversary hands-on workshop

Masatoshi SEKI

introduction _____ 5min.

workshop

1. Hello, World. _____ 20min.
2. Key value store _____ 20min.
3. Queue _____ 20min.

closing _____ 5min.

What is dRuby?

- Distributed Object System
- Can invoke methods in different process
- Can send objects between process
- Pure Ruby

The dRuby Book (web edition)

`http://www.druby.org/sidruby/`

1. Hello, World.

Learn dRuby setup and Remote Method Invocation.

➡ list 1-1 hello_server.rb

```
require 'drb'

class Hello
  def greeting
    puts('Hello, World.')
  end
end

uri = 'druby://localhost:54000'
DRb.start_service(uri, Hello.new)
sleep
```

Bind a URI to an object, and start the dRuby server thread.

It's a server, so don't let it finish

➡ list 1-2 hello_client.rb

```
require 'drb'

DRb.start_service
uri = 'druby://localhost:54000'
it = DRbObject.new_with_uri(uri)
it.greeting
```

Start the dRuby server thread,
Create a proxy object,
and send a message.

Step 1.
Run hello_server.rb on terminal 1.

➡ terminal 1

```
$ ruby hello_server.rb
```

Step 2.
Run hello_client.rb on terminal 2.

➡ terminal 2

```
$ ruby hello_client.rb
```

Then, "Hello, World." is displayed on terminal 1.

Q. Terminate the server then run the client. What happens?

2. Key value store

Learn to exchange objects. This exercise uses 3 terminals. Output of irb is omitted.

Step 1.
Start the hash server.

➡ terminal 1

```
$ irb -r drb --simple-prompt
>> kvs = Hash.new
>> uri = 'druby://localhost:54320'
>> DRb.start_service(uri, kvs)
```

Step 2.
Store objects into the hash server.

➡ terminal 2

```
$ irb -r drb --simple-prompt
>> DRb.start_service
>> uri = 'druby://localhost:54320'
>> kvs = DRbObject.new_with_uri(uri)
>> kvs['greeting'] = 'hello, world'
>> kvs['stdout'] = $stdout
=> #<IO:<STDOUT>>
```

Step 3.
Retrieve objects.

➡ terminal 3

```
$ irb -r drb --simple-prompt
>> DRb.start_service
>> uri = 'druby://localhost:54320'
>> kvs = DRbObject.new_with_uri(uri)
>> kvs['greeting']
=> "hello, world"
>> kvs['stdout']
=> #<DRb::DRbObject:0x00... @uri="druby://172.17.0.3:46761", @ref=...>
```

Q. Let's store various kind object (eg. String, Integer, Hash, IO).

Q. What happens when kvs['stdout'].puts('hello, again') ?

3. Queue

Synchronize process using queue.

Step 1.
Start the queue server.

➡ terminal 1

```
$ irb -r drb --simple-prompt
>> DRb.start_service('druby://localhost:54321', SizedQueue.new(1))
```

Step 2.
Run the consumer and the producer.

➡ terminal 2

```
$ irb -r drb --simple-prompt
>> DRb.start_service
>> uri = 'druby://localhost:54321'
>> queue = DRbObject.new_with_uri(uri)
```

➡ terminal 3

```
$ irb -r drb --simple-prompt
>> DRb.start_service
>> uri = 'druby://localhost:54321'
>> queue = DRbObject.new_with_uri(uri)
```

Step 3.
Let's coordinate two processes.

➡ terminal 2 (cont.)

```
>> queue.pop
=> "one"
>> queue.pop
=> 2.0
>> queue.pop
        (* Block! queue is empty. *)
=> 3
>> queue.pop
        (* Block! queue is empty. *)
=> :four
```

➡ terminal 3 (cont.)

```
>> queue.push("one")
>> queue.push(2.0)
        (* Block! queue is full. *)


>> queue.push(3)


>> queue.push(:four)
```

Q. Let's increase consumer, producer.

Q. Let's increase size of queue.

Q. Let's store various kind object (eg. String, Integer, Hash, IO).

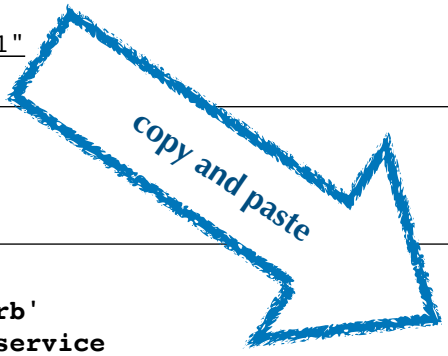
4. Docker

Container to Container communication.

Step 1.

➡ terminal 1

```
$ docker run -it ruby
irb(main):001:0> require 'drb'
irb(main):002:0> DRb.start_service('druby://:54321', {})
irb(main):003:0> DRb.front
=> {}
irb(main):004:0> DRb.uri
=> "druby://172.17.0.2:54321"
irb(main):005:0>
```



Step 2.

➡ terminal 2

```
$ docker run -it ruby
irb(main):001:0> require 'drb'
irb(main):002:0> DRb.start_service
irb(main):003:0> kvs = DRbObject.new_with_uri("druby://172.17.0.2:54321")
irb(main):004:0> kvs['stdout'] = $stdout
=> #<IO:<STDOUT>>
irb(main):005:0>
```

Step 3.

➡ terminal 1 (cont.)

```
irb(main):005:0> DRb.front
=> {"stdout"=>#<DRb::DRbObject:0x00... @uri="druby://172.17.0.3:46761",
@ref=...>}
irb(main):006:0> DRb.front['stdout'].puts("hello, again")
=> nil
irb(main):007:0>
```

➡ terminal 2 (cont.)

```
irb(main):005:0> hello, again
```

5. Docker and Ring

Finding a service with Ring.

Step 1.

➡ terminal 1

Run a name server.

```
$ docker run -it ruby
irb(main):001:0> require 'rinda/ring'
irb(main):002:0> require 'rinda/tuplespace'
irb(main):003:0> DRb.start_service
irb(main):004:0> place = Rinda::RingServer.new(Rinda::TupleSpace.new)
irb(main):005:0>
```

Step 2.

➡ terminal 2

Find a name server, and retrieve the queue server.

```
$ docker run -it ruby
irb(main):001:0> require 'rinda/ring'
irb(main):002:0> DRb.start_service
irb(main):003:0> ns = Rinda::RingFinger.primary
irb(main):004:0> _, _, queue, _ = ns.read([:name, :queue, DRbObject, nil])
                                                                  (* Block!*)
```

Step 3.

➡ terminal 3

Run a queue service provider.

```
$ docker run -it ruby
irb(main):001:0> require 'rinda/ring'
irb(main):002:0> DRb.start_service
irb(main):003:0> queue = SizedQueue.new(1)
irb(main):004:0> ns = Rinda::RingFinger.primary
irb(main):005:0> ns.write([:name, :queue, queue, "queue"])
irb(main):006:0> queue.pop
                                                                  (* Block!*)
```

➡ terminal 2 (cont.)

```
=> #<DRb::DRbObject:0x0...>
irb(main):005:0> queue.push("hello, again")
irb(main):006:0>
```

➡ terminal 3 (cont.)

```
=> "hello, again"
irb(main):007:0>
```

Slide Sponsors

手作りのキャンピングカーで、生活して、仕事して、そして、とことん遊んでやろうと思っているtoRubyファンの青木です
製作中ですけど、高速道路で壊れていなければ、ここRubykaigi（福岡）にも栃木から来ている予定です
SketchUpという3D Cadがきっかけでに興味を持ち始めた土木工事の現場監督ですが、 よろしくお願いします

Aoki Shinichi

20周年おめでとうございます!

@hisashim

現場Rails、おかげさまで3刷です！ありがとうございます〜！！！

@tatsuoSakurai

「Ruby超入門」「Railsの教科書」「RubyとRailsの学習ガイド」発売中！

@igaiga555

みなさん、関さんのdRubyの話、『n月刊ラムダノート』の記事として読みたいですよね！

@golden_lucky

今年もsekiトークをお楽しみください！とちぎRubyKaigiも皆さんぜひー。なお、例年通り、テレビを買い換える人はご相談ください。

@tsuboi

20周年おめでとうございます。とちぎRuby会議08は6/29です。遊びに来てください。

@vestige_

20周年おめでとうございます!

@arika

dRuby 20th🎉

@tricknotes

About Me

Masatoshi SEKI, www.druby.org, @m_seki, seki, SW-7603-4893-2503, pokémon GO 3600 2703 1425, ポケ森 8436 2321 777, https://t.co/oramJa8g0p