How to test code with mruby

mruby testing in the wild

GMOペパボ株式会社



self.introduce



=>
{
  name: “SHIBATA Hiroshi”,
  nickname: “hsbt”,
  title: “Chief engineer at GMO Pepabo, Inc.”,
  commit_bits: [“ruby”, “rake”, “rubygems”, “rdoc”, “tdiary”, “hiki”, “railsgirls”, “railsgirls-jp”, “jenkins”],
  sites: [“ruby-lang.org”, “rubyci.com”, “railsgirls.com”, “railsgirls.jp”],
}

Today’s target

- Web engineer(Rails programmer)
- Operation engineer
- QA/test engineer
- mruby committer

Test Everything

- Application
  - xUnit, BDD testing tool, End-to-End testing tool
- Middleware
  - (nothing)
- Server with IaaS
  - Serverspec, Infrataster

mruby

What's mruby?

“mruby is the lightweight implementation of the Ruby language complying to (part of) the ISO standard. Its syntax is Ruby 1.9 compatible.”

https://github.com/mruby/mruby#whats-mruby

Differences between mruby and CRuby

- The mruby runtime and libraries are embedded all into a single binary.
- By default, mruby provides just a minimum set of standard libraries such as String, Array, Hash, etc.
- Some of standard libraries in CRuby are NOT ones in mruby, for example, IO, Regex, Socket, etc..
- mruby doesn't provide “require”, “sleep”, “p”, etc.

Advantages of mruby

- Single binary without pure ruby files.
- Embeddable into middlewares like below:
  - apache/nginx
  - groonga
  - mysql
- Fun!!1 # most important thing

ngx_mruby

Introduction to ngx_mruby

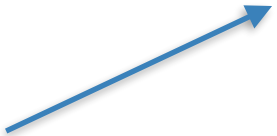
“ngx_mruby is A Fast and Memory-Efficient Web Server Extension Mechanism Using Scripting Language mruby for nginx.”

https://github.com/matsumoto-r/ngx_mruby#whats-ngx_mruby

```
location /hello {
    mruby_content_handler /path/to/hello.rb cache;
}
```



In “nginx.conf”!!!



```
location /proxy {
    mruby_set_code $backend '
        backends = [
          "test1.example.com",
          "test2.example.com",
          "test3.example.com",
        ]
        backends[rand(backends.length)]
    ';
}
```

How to build ngx_mruby (and mruby)

I suggest to try it on OS X or Linux environment. You can change embedded mgem via “build_config.rb” in ngx_mruby. repository.

```
$ git clone https://github.com/matsumoto-r/ngx_mruby
$ git clone https://github.com/nginx/nginx
$ cd ngx_mruby
$ git submodule init && git submodule update

comment-out mruby-redis and mruby-vedis

$ ./configure —with-ngx-src-root=../nginx
$ make build_mruby
$ make
$ cd ../nginx
$ ./objs/nginx -V
```

Run ruby code with ngx_mruby

- mruby_content_handler/mruby_content_handler_code
- mruby_set/mruby_set_code
- mruby_init/mruby_init_code
- mruby_init_worker/mruby_init_worker_code

Sample code of ngx_mruby

```
class ProductionCode
  def initialize(r, c)
    @r, @c = r, c
  end

  def allowed_ip_addresses
    %w[
      128.0.0.1
    ]
  end

  def allowed?
    if (allowed_ip_addresses & [@c.remote_ip, @r.headers_in['X-Real-IP'], @r.headers_in['X-Forwarded-
For']].compact).size > 0
      return true
    end
    (snip for memcached)
  end
 return false
end
ProductionCode.new(Nginx::Request.new, Nginx::Connection.new).allowed?
```

Sample configuration of nginx

```
location /path {

  mruby_set $allowed ‘/etc/nginx/handler/production_code.rb' cache;

  if ($allowed = 'true'){
    proxy_pass http://upstream;
  }

  if ($allowed = 'false'){
    return 403;
  }
}
```

Usecase of ngx_mruby

- Calculation of digest hash for authentication.
- Data sharing with Rails application.
- To replace ugly complex nginx.conf with clean, simple, and TESTABLE ruby code.

Middleware as a Code

Advanced topic of ngx_mruby

- Development process
- Continuous Integration
- Test
- Deployment

We'll focus only on testing only

Testing code of mruby

What's motivation

- We are using ngx_mruby in production.
- We should test every production code.
- Testing mruby is a cutting edge technical issue.

Sample code of ngx_mruby

```
class ProductionCode
  def initialize(r, c)
    @r, @c = r, c
  end

  def allowed_ip_addresses
    %w[
      128.0.0.1
    ]
  end

  def allowed?
    if (allowed_ip_addresses & [@c.remote_ip, @r.headers_in['X-Real-IP'], @r.headers_in['X-Forwarded-
For']].compact).size > 0
      return true
    end
    (snip for memcached)
  end
 return false
end
ProductionCode.new(Nginx::Request.new, Nginx::Connection.new).allowed?
```

Prototype concept

- Use CRuby(version independent: 2.0.0, 2.1, 2.2)
- Use test-unit
- Test “ruby code” without real world behavior.

Dummy class of ngx_mruby

```
class Nginx
  class Request
    attr_accessor :uri, :headers_in, :args, :method, :hostname
    def initialize
      @uri = nil
      @headers_in = {}
      @args = nil
      @method = 'GET'
      @hostname = nil
    end
  end

  class Connection
    attr_accessor :remote_ip
    def initialize
      @remote_ip = nil
    end
  end
```

Skeleton of test-case

```
require_relative '../lib/production/code/path/mruby.rb'

class MRubyTest < Test::Unit::TestCase
  def setup
    @r = Nginx::Request.new
    @c = Nginx::Connection.new
  end

  def test_discard_access
    assert !ProductionCode.new(@r, @c).allowed?
  end
end
```

Permit specific requests with IP address

```
require_relative '../lib/production/code/path/mruby.rb'

class MRubyTest < Test::Unit::TestCase
  def setup
    @r = Nginx::Request.new
    @c = Nginx::Connection.new
  end

  def test_ip_access
    @c.remote_ip = '128.0.0.1'
    @r.uri = '/secret/file/path'
    assert ProductionCode.new(@r, @c).allowed?
  end
end
```

Run test

```
% ruby test/production_code_test.rb
Loaded suite test/production_code_test
Started
.........

Finished in 0.031017 seconds.
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------
9 tests, 15 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifications
100% passed
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------
------------------
290.16 tests/s, 483.61 assertions/s
```

We can test it!

Testing requirements

- Environment
  - OS and Architecture
  - Library
  - Middleware
- Code
- Input

Our concerns on CRuby testing

- We can test “ruby code”. But it’s not fulfill testing requirements. We need to test ngx_mruby behavior.
- We use a lot of mock/stub classes. It’s ruby’s dark-side.
- We need to make easy task runner.

Testing code of mruby using mruby

Use mruby directly instead of CRuby

mruby-mtest

build_config.rb

```
MRuby::Build.new do |conf|

  (snip)

  conf.gem :github => 'matsumoto-r/mruby-uname'

  # ngx_mruby extended class
  conf.gem '../mrbgems/ngx_mruby_mrblib'

  con.gem :github => 'iij/mruby-mtest'

  (snip)
end
```

test_4m_test.rb

```
class Test4MTest < MTest::Unit::TestCase
  def test_assert
    assert(true)
    assert(true, 'true sample test')
  end
end

MTest::Unit.new.run
```

Inline testing for mruby-mtest

```
class ProductionCode
  (snip)
end

if Object.const_defined?(:MTest)
  class Nginx
    (snip)
  end

  class ProductionCode < MTest::Unit::TestCase
    (snip)
  end

  MTest::Unit.new.run
else
  ProductionCode.new(Nginx::Request.new, Nginx::Connection.new).allowed?
end
```

Build mruby for mruby testing

You need to get mruby binary before embed ngx_mruby.

```
$ cd ngx_mruby/mruby
$ cp ../build_config.rb .
$ make
$ cp bin/mruby /path/to/test/bin
```

Test runner for mruby-mtest

```
require 'rake'

desc 'Run mruby-mtest'
task :mtest do
  target = "modules/path/to/production/code"
  mruby_binary = File.expand_path("../#{target}/test_bin/mruby", __FILE__)
  mruby_files = FileList["#{target}/**/*.rb"]
  mruby_files.each do |f|
    absolute_path = File.expand_path("../#{f}", __FILE__)
    system "#{mruby_binary} #{absolute_path}"
  end
end
```

Advantage of mruby testing

Rapid!

```
% rake mtest

# Running tests:

.........

Finished tests in 0.007924s, 1135.7900 tests/s, 1892.9833 assertions/s.

9 tests, 15 assertions, 0 failures, 0 errors, 0 skips
```

Advantage of mruby testing

Direct use of mruby binary

```
% ./modules/nginx_app_proxy/test_bin/mruby -v
mruby 1.1.0 (2014-11-19)
^C
```

Next challenge

- mruby binary can have different library from one in production.
- For continuous integration, we need to prepare cross-compile or live compile environment.
- Replace nginx.conf with mruby code backed by test code.

[RDRC 2015] [search]



Hiroshi Shibata
Chief Engineer, GMO Pepabo, Inc.

HTTP Programming with mruby

mruby is the lightweight implementation of the Ruby language and was released about a year ago. Can we use mruby to write web services?

This answer is YES - our company used mruby in large scaled web services. Even with mruby, we were able to create web services with tests and gems, and it also helped to solve some problems using Ruby code outside of a Rails application. In essence, mruby also provides programming features like HTTP to us web programmers.

Speaker's Bio

CRuby committer and root operation engineer of ruby-lang.org. I am a full-stack developer at GMO Pepabo.

We are hiring!!1

募集中の職種一覧

※ご応募前に必ず 採用応募者のみなさまの個人情報取扱いについて のご一読をお願いいたします。

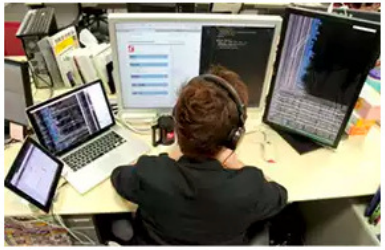
オープンポジション

職種名	勤務地
オープンポジション (正社員)	東京

エンジニア

- WEBアプリケーションエンジニア
- モバイルアプリケーションエンジニア
- インフラエンジニア
- 情報システムエンジニア

エンジニアの働き方 ›



ペパランチョン ›



GMOペパボ株式会社