mason gem
January 2016

Goal: I want to make a gem that has all my shared ruby code, so that I can easily use it with every Ruby project I do. Since I have never built a Ruby gem before, I will document the process, as I am wont to do.

Work Log:

2016-01-23: add CommandWrapper utility class for running external commands
2016-01-23: add some unit tests
2016-01-23: add a module to namespace stuff (Mama took the kids to the doctor—now's my chance!)
2016-01-09: figure out how to conveniently work on the gem and a project using it at the same time
2016-01-09: figure out how to use the gem by referencing its GitHub repo
2016-01-09: create My Little Pony® Baby's First Gem™

Links:

RubyGems guide to how to make your own gem (useful)

Ways to specify a local gem in Gemfile

以上

add a module to namespace stuff
2016-01-23

OK, in our last episode, the Mason gem just had a single file: **lib/mason.rb**

Now I want to add a bunch of useful classses and things to the gem. But I want to keep those all under the "Mason" namespace. The main (only?) way to do that in ruby is to put things in a module (the monk can tell you more about this). So I want to make a "Mason" module, into which I will (someday) put a bunch of awesomeness.

Remember, that is the root file: "The convention is to have one Ruby file with the same name as your gem, since that gets loaded when require 'gemname' is run. That one file is in charge of setting up your gem's code and API."

As explained in The Fine Documentation, when you add a bunch of files to your gem, that root file is responsible for **require**ing them, and also they should (by convention) live in **lib/*gemname***. So in this case, when we add a new ruby file for the CommandWrapper class, the tree ends up like this:

```
[mason@tokyo mason-gem (master)]$ tree
.
├── README.md
├── lib
│   ├── mason                        ← new!
│   │   └── command_wrapper.rb       ← new!
│   └── mason.rb
├── mason.gemspec
```

And then the root file, **mason.rb**, requires it like this:

```
[mason@tokyo mason-gem (master)]$ pygmentize lib/mason.rb

require 'mason/command_wrapper'

module Mason

  class Mason # we also moved the old Mason class into the new Mason module.

    def boogie
      puts "Sorry, the boogie feature has been removed in this new version."
    end

  end

end

[mason@tokyo mason-gem (master)]$
```

For now there is no real code in **command_wrapper.rb**, as I just want to confirm the gem structure is set up all right:

```
[mason@tokyo mason-gem (master)]$ pygmentize lib/mason/command_wrapper.rb
```

```ruby
module Mason

  class CommandWrapper

    def run
      nope = "nope nope!"
      puts nope
      return nope
    end

  end

end
```

[mason@tokyo mason-gem (master)]$

OK, so is it?

Update the test script to reflect the changes to the gem:

[mason@tokyo test-the-gem-bro (master)]$ pygmentize test.rb

```ruby
#! /usr/bin/env ruby

require 'mason'

foo = Mason::Mason.new

puts "Here's a Mason instance: #{foo}"

puts "Will it boogie?"

foo.boogie

bar = Mason::CommandWrapper.new
bar.run
```

[mason@tokyo test-the-gem-bro (master)]$

And, because Bundler has bundled with the old version of the gem, we need to do a **bundle update**:

```
[mason@tokyo mason-gem (master)]$ cd test-the-gem-bro
[mason@tokyo test-the-gem-bro (master)]$
[mason@tokyo test-the-gem-bro (master)]$
[mason@tokyo test-the-gem-bro (master)]$ bundle update
Resolving dependencies...
Using mason 0.0.2 (was 0.0.1) from source at ../
Using bundler 1.7.7
Your bundle is updated!
[mason@tokyo test-the-gem-bro (master)]$
```

Et voila!

```
[mason@tokyo test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason::Mason:0x007ff504b0b188>
Will it boogie?
Sorry, the boogie feature has been removed in this new version.
nope nope!
[mason@tokyo test-the-gem-bro (master)]$
```

In this case, "nope nope" means "yep it worked".

add CommandWrapper utility class for running external commands
2016-01-23

The motivation this morning was the annoying proces of installing rbenv and ruby and all the extra steps required to make it work will with Japanese text on OS X. (One might think that if anything was gonna just work with Japanese out of the box, it would be ruby on OS X... but one would be wrong.)

Anyway I want to automate that process so that I never have to do it manually again. (especially the searching Google for how to do it part).

So, let's bring our old pal CommandWrapper into mason-gem land. I think it is probably OK(?) to just have all the test cases live in **test** and not try to mimic the folder structure we have in **lib**.

So to start, I will just copy and paste the existing test cases into test/ test_command_wrapper:

```
[mason@tokyo mason-gem (master)]$ pygmentize test/test_command_wrapper.rb
require 'minitest/autorun'
require 'mason/command_wrapper'

class CommandWrapperTests < MiniTest::Unit::TestCase
  # mason 2016-01-23: bringing the first bit of my standard ruby code under one
roof...
  # mason 2011-08-05 fucking around porting ancient python stuff

  def setup
    @class    = Mason::CommandWrapper
    @instance = Mason::CommandWrapper.new
  end


  def test_instantiation_without_command
    # A wrapper should have no command by default, and trying to run it should
raise.

    assert_nil @instance.command
    assert_raises(Mason::CommandWrapper::InvalidCommandError) {@instance.run}
  end


  def test_stdout
    @instance = @class.new 'echo "おはようございます"'
    @instance.run

    assert @instance.stdout.include? "おはよう"
    assert_equal 0, @instance.exit_status
  end


  def test_stderr
    @instance = @class.new 'ls /bonkGRONKdonkDONK~~NONEXISTENT'
    @instance.raise_on_bad_exit = false
```

```ruby
    @instance.run

    assert @instance.stderr.include? "ls:" # because any message might be localized
    refute_equal 0, @instance.exit_status
  end


  def test_simple_ls_use_case
    @instance.command = "ls -la /"
    @instance.run

    assert_equal 0, @instance.exit_status
    assert @instance.stdout.include? "drwx"
      # surely this will be there on any system we might use?
    assert @instance.stderr == ""
    assert @instance.exit_status == 0
  end


  def test_working_dir_is_picked_up_at_instantiate_time
    assert_equal @instance.working_directory, Dir.pwd

    Dir.chdir("/Users")
    @instance = Mason::CommandWrapper.new
    assert_equal @instance.working_directory, "/Users"
  end


  def test_exit_status
    @instance.raise_on_bad_exit = false
    assert_nil @instance.exit_status
    @instance.working_directory = "/Users"
    @instance.command = "ls"
    @instance.run
    assert_equal 0, @instance.exit_status

    @instance.command = "ls DOESNTEXISTfjdhsfldjkhlhdfsjk"
    @instance.run
    assert @instance.exit_status != 0
  end


  def test_run_command_return_val
    exit_status, stdout_output, stderr_output = @instance.run_command 'ls -l /'
    assert exit_status == 0
    assert stdout_output.length > 0
    assert stderr_output.length == 0
  end


  def test_should_raise_on_nonzero_exit_status
    @instance = @class.new 'ls -la'
    @instance.run
    assert_equal 0, @instance.exit_status

    @instance = @class.new('ls -la /dshjkflashdfklahfhsadjkfhlasdhfhasdlhfu')
    assert_raises(Mason::CommandWrapper::UnexpectedExitStatusError) {@instance.run}

    @instance.raise_on_bad_exit = false
    @instance.run
```

```
    assert @instance.exit_status != 0
  end

end
[mason@tokyo mason-gem (master)]$
```

Of course, they will all fail:

```
[mason@tokyo mason-gem (master)]$ rake test
Run options: --seed 2157

# Running tests:

EEEEEEEE.

Finished tests in 0.001030s, 8737.8641 tests/s, 970.8738 assertions/s.

  1) Error:
CommandWrapperTests#test_simple_ls_use_case:
NoMethodError: undefined method `command=' for #<Mason::CommandWrapper:
0x007fbf2c8cfcd8>
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:42:in
`test_simple_ls_use_case'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  2) Error:
CommandWrapperTests#test_working_dir_is_picked_up_at_instantiate_time:
NoMethodError: undefined method `working_directory' for #<Mason::CommandWrapper:
0x007fbf2c8cf1e8>
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:54:in
`test_working_dir_is_picked_up_at_instantiate_time'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
```

```
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  3) Error:
CommandWrapperTests#test_run_command_return_val:
NoMethodError: undefined method `run_command' for #<Mason::CommandWrapper:
0x007fbf2c8ce798>
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:77:in
`test_run_command_return_val'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
```

```
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  4) Error:
CommandWrapperTests#test_stdout:
ArgumentError: wrong number of arguments (1 for 0)
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:23:in `initialize'
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:23:in `new'
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:23:in `test_stdout'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  5) Error:
CommandWrapperTests#test_should_raise_on_nonzero_exit_status:
ArgumentError: wrong number of arguments (1 for 0)
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:85:in `initialize'
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:85:in `new'
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:85:in
`test_should_raise_on_nonzero_exit_status'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
```

```
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  6) Error:
CommandWrapperTests#test_instantiation_without_command:
NoMethodError: undefined method `command' for #<Mason::CommandWrapper:
0x007fbf2c8ccbf0>
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:17:in
`test_instantiation_without_command'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  7) Error:
CommandWrapperTests#test_exit_status:
NoMethodError: undefined method `raise_on_bad_exit=' for #<Mason::CommandWrapper:
0x007fbf2c8cc2b8>
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:63:in
`test_exit_status'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
```

```
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

  8) Error:
CommandWrapperTests#test_stderr:
ArgumentError: wrong number of arguments (1 for 0)
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:32:in `initialize'
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:32:in `new'
    /Users/mason/Code/mason-gem/test/test_command_wrapper.rb:32:in `test_stderr'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1265:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:940:in
`block in _run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:933:in
`_run_suite'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`block in _run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`map'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:914:in
`_run_suites'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:884:in
`_run_anything'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1092:in
`run_tests'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1079:in
`block in _run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`each'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1078:in
`_run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:1066:in
`run'
    /usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/minitest/unit.rb:802:in
`block in autorun'

9 tests, 1 assertions, 0 failures, 8 errors, 0 skips
```

```
rake aborted!
Command failed with status (1): [ruby -I"lib:test" -I"/usr/local/var/rbenv/
versions/2.1.5/lib/ruby/2.1.0" "/usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/
rake/rake_test_loader.rb" "test/test*.rb" ]

Tasks: TOP => test
(See full trace by running task with --trace)
[mason@tokyo mason-gem (master)]$
```

Hmm, LOL, one actually did not fail? That's pretty interesting, since the class being tested currently looks like this:

```
[mason@tokyo mason-gem (master)]$ pygmentize lib/mason/command_wrapper.rb
module Mason

  require 'open3'


  class CommandWrapper

    def run
      nope = "nope nope!"
      puts nope
      return nope
    end

    class InvalidCommandError < StandardError
    end

    class UnexpectedExitStatusError < StandardError
    end

  end # CommandWrapper class

end # Mason module
[mason@tokyo mason-gem (master)]$
```

But anyway, no time for that. Implement the class:

```
[mason@tokyo mason-gem (master)]$ pygmentize lib/mason/command_wrapper.rb
module Mason

  require 'open3'


  class CommandWrapper

    attr_accessor :command, # the whole command string with args, e.g. 'ls -al /
foo'
                  :working_directory,
                  :expected_exit_status,
                  :raise_on_bad_exit,
                  :exit_status,
                  :stdout,
                  :stderr,
                  :start_time,
                  :end_time
```

```ruby
    def initialize(command = nil)
      @command            = command
      @working_directory  = Dir.pwd
      @expected_exit_status = 0
      @raise_on_bad_exit  = true
      @exit_status        = nil
      @stdout_output      = nil
      @stderr_output      = nil
      @start_time         = nil
      @end_time           = nil

    end

    def run()
      @exit_status, @stdout, @stderr, @start_time, @end_time =
run_command(@command)

      if @raise_on_bad_exit && (@exit_status != @expected_exit_status)
        raise UnexpectedExitStatusError.new("#{to_s :short}: expected
#{@expected_exit_status} but got #{@exit_status}")
      end
    end


    def run_command(cmd_str)
      # run cmd in the shell, wait for exit, and return exit_status, stdout_output,
stderr_output, start_time, end_time
      # e.g. exit_status, stdout_output, stderr_output = run_command 'ls -l /'

      if [nil, ''].include? cmd_str
        raise InvalidCommandError.new("invalid command: #{cmd_str}")
      end

      start_time = Time.now

      # Mason 2010-10-17: here is one way to run a one shot command line tool
      # subprocess in Ruby 1.9. The read calls block until the subprocess exits,
      # so when you want get fancy and communicate with the subprocess then
      # this isn't what you need. Also, there's no mechanism yet to have a timeout
or
      # interrupt a task that is e.g. stuck waiting for input or otherwise
infinite.

      stdin, stdout, stderr, wait_thr = Open3.popen3(cmd_str)
      # pid = wait_thr[:pid]  # pid of the started process.
      # puts "Execute command: " + cmd
      # puts "pid: " + pid.to_s

      stdout_output = stdout.read
      stderr_output = stderr.read
        # those block until exit, so after this line subprocess is done

      status = wait_thr.value  # Process::Status object returned.

      stdin.close  # per docs, stdin, stdout and stderr should be closed in this
form.
      stdout.close # dunno if that is really necessary in this use case but it does
      stderr.close # not seem to hurt...
```

```ruby
      exit_status = status.exitstatus
      end_time    = Time.now

      return exit_status, stdout_output, stderr_output, start_time, end_time
    end

    def self.run_command(cmd_str)
      cw = self.new cmd_str
      cw.run
    end


    def to_s(style = :short)
      "#{self.class.name}: #{@command}"
    end

    # def run
    #   nope = "nope nope!"
    #   puts nope
    #   return nope
    # end

    class InvalidCommandError < StandardError
    end

    class UnexpectedExitStatusError < StandardError
    end

  end # CommandWrapper class

end # Mason module
[mason@tokyo mason-gem (master)]$
```

And, ba-da-boom:

```
[mason@tokyo mason-gem (master)]$ rake test
Run options: --seed 30642

# Running tests:

.........

Finished tests in 0.030348s, 296.5599 tests/s, 724.9242 assertions/s.

9 tests, 22 assertions, 0 failures, 0 errors, 0 skips
[mason@tokyo mason-gem (master)]$
```

OK, so we are still near-zero in terms of actual utility, BUT we have a gem which contains multiple classes and unit tests. This should be all the prep work I need to do before making this gem actually do something useful to me -- install modern Ruby and rbenv and make it work with Japanese on this Mac. (As you can see in some of the printouts above, this Mac still has old Ruby 2.1.... that's because I am waiting for this gem to gain the capability of updating it for me.)

Well the current time is Miller Time, so .... until next time!

add some unit tests
2016-01-23

My next goal here is to add some actual useful code to this gem. So let's add some unit tests, as a first step.

I think the normal way people do this is to add to the gem's root folder:
- a Rakefile
- a "tests" subfolder
- a set of 1 or more test case source files inside the "tests" subfolder

So, I do all that:

```
[mason@tokyo mason-gem (master)]$ tree
.
├── README.md
├── Rakefile
├── lib
│   ├── mason
│   │   └── command_wrapper.rb
│   └── mason.rb
├── mason.gemspec
├── test
│   └── test_mason.rb
```

I just use the default Rakefile from the guide for now:

```
[mason@tokyo mason-gem (master)]$ pygmentize Rakefile

# Mason 2016-01-23: I got this default started Rakefile from http://
guides.rubygems.org/make-your-own-gem/#writing-tests

require 'rake/testtask'

Rake::TestTask.new do |t|
  t.libs << 'test'
end

desc "Run tests"
task :default => :test

[mason@tokyo mason-gem (master)]$
```

And add a unit test file (one that will fail, to start with):

```
[mason@tokyo mason-gem (master)]$ pygmentize test/test_mason.rb

require 'minitest/autorun'
require 'mason'

class MasonTest < Minitest::Unit::TestCase

  def test_sanity
    assert_equal "🐷", "your mom"
```

```
      end

  end
```

```
[mason@tokyo mason-gem (master)]$
```

...et voilà!

```
[mason@tokyo mason-gem (master)]$ rake test
Run options: --seed 37958

# Running tests:

F

Finished tests in 0.006439s, 155.3036 tests/s, 155.3036 assertions/s.

  1) Failure:
MasonTest#test_sanity [/Users/mason/Code/mason-gem/test/test_mason.rb:7]:
Expected: "🐷"
  Actual: "your mom"

1 tests, 1 assertions, 1 failures, 0 errors, 0 skips
rake aborted!
Command failed with status (1): [ruby -I"lib:test" -I"/usr/local/var/rbenv/
versions/2.1.5/lib/ruby/2.1.0" "/usr/local/var/rbenv/versions/2.1.5/lib/ruby/2.1.0/
rake/rake_test_loader.rb" "test/test*.rb" ]

Tasks: TOP => test
(See full trace by running task with --trace)
[mason@tokyo mason-gem (master)]$
```

Cool! In this case, the "F" is for "Fuck yeah, we are almost to the point where we can add the first useful code to this gem."

create My Little Pony® Baby's First Gem™
2016-01-09

Minimally, it is just 2 files in a simple hierarchy:

```
[mason@MacBook-Pro-No mason-gem]$ tree
.
├── lib
│   └── mason.rb
└── mason.gemspec
```

The gem spec can apparently be complicated, according to the docs, but this simple one suffices for now:

```
[mason@MacBook-Pro-No mason-gem]$ pygmentize mason.gemspec

Gem::Specification.new do |s|
  s.name        = 'mason'
  s.version     = '0.0.1'
  s.date        = '2016-01-08'
  s.summary     = "Mason!"
  s.description = "A personal utility gem by and for Mason."
  s.authors     = ["Mason Mark"]
  s.email       = 'mason@masonmark.com'
  s.files       = ["lib/mason.rb"]
  s.homepage    = 'http://masonmark.com'
  s.license     = 'MIT'
end

[mason@MacBook-Pro-No mason-gem]$
```

...and the code itself:

```
[mason@MacBook-Pro-No mason-gem]$ pygmentize lib/mason.rb

class Mason

  def boogie
    puts "w00t w00t"
  end

end

[mason@MacBook-Pro-No mason-gem]$
```

The gem spec and source code are ready; next, build the gem:

```
[mason@MacBook-Pro-No mason-gem]$ gem build mason.gemspec
  Successfully built RubyGem
  Name: mason
  Version: 0.0.1
  File: mason-0.0.1.gem
[mason@MacBook-Pro-No mason-gem]$


[mason@MacBook-Pro-No mason-gem]$ ls -l
```

```
drwxr-xr-x  3 mason  staff   102 Jan  8 20:58 lib
-rw-r--r--  1 mason  staff  4096 Jan  9 10:03 mason-0.0.1.gem
-rw-r--r--@ 1 mason  staff   378 Jan  9 09:52 mason.gemspec
[mason@MacBook-Pro-No mason-gem]$
```

Cool! The gem is built.

Now how can we use it? Well, if we were lame, we could install it globally on the local machine like this:

```
[mason@MacBook-Pro-No mason-gem]$ sudo gem install ./Mason-0.0.1.gem
Password:
Successfully installed mason-0.0.1
Parsing documentation for mason-0.0.1
Installing ri documentation for mason-0.0.1
1 gem installed
[mason@MacBook-Pro-No mason-gem]$
```

And then use it like this:

```
[mason@MacBook-Pro-No mason-gem]$ irb
irb(main):001:0> require 'mason'
=> true
irb(main):002:0> foo = Mason.new
=> #<Mason:0x007fb1a2a253b8>
irb(main):003:0> foo.boogie
w00t w00t
=> nil
irb(main):004:0> ^D
[mason@MacBook-Pro-No mason-gem]$
```

But that's not what we are trying to accomplish, so:

```
[mason@MacBook-Pro-No mason-gem]$ sudo gem uninstall Mason
Password:
Successfully uninstalled Mason-0.0.1
[mason@MacBook-Pro-No mason-gem]$
```

What we actually want to do is require this gem in a Gemfile used with Bundler. The good and easy way to do that is to put it on my own Github, since Bundler has built-in support for referencing GitHUb-hosted gems in the Gemfile.

But even easier at this stage (since this gem isn't yet on GitHub) is just requiring it locally via absolute path. Here is the working setup with a throwaway test project:

The minimum setup to use the gem in isolation via Bundler is a Gemfile to specify the

gem, and a ruby script to run.

```
"mason"  path  "../mason-gem/"
# path is to parent dir, not gem itself
```

For this exercise, we've specified only the mason gem. Then we require it in the source code for our project:

```
#! /usr/bin/env ruby

require 'mason'

     Mason
puts "Here's a Mason instance: #{   }"
puts "Will it boogie?"
```

Next, run it with Bundler. It seems that no 'bundle install' command is even necessary with this minimal setup.

Boogie. Of course, it won't run without Bundler, because the gem is not installed on the local system:

That's as it should be.

For my next stupendous trick, I think I will put this gem on Github and then use it from

there.

figure out how to conveniently work on the gem and a project using it at the same time

2016-01-09

Oh, cool. If using the local path way of referencing the gem in the Gemfile, it looks like you don't even have to actually do a `gem build`.

Change the Gemfile to reference the relative path to the gem's source code folder:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ pygmentize -l ruby Gemfile
# gem "mason", github: "masonmark/mason-gem"
  # When not actively working on the gem, referencing it via github is good.
  # However, when working on the gem itself, it's more covenient to reference it
  # via local path, because then there's no need to push the gem code to github
  # and then have to 'bundle install' in the project using it.

gem "mason", path: "../"
  # path is to parent dir, not gem itself

[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Note that the built gem is not present:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$  ls -l ..
total 1392
drwxr-xr-x  8 mason   staff      272 Jan  9 12:17 README-mason-gem
-rw-r--r--@ 1 mason   staff   701669 Jan  9 12:17 README-mason-gem.pdf
drwxr-xr-x  6 mason   staff      204 Jan  9 12:17 README-mason-gem.vpdoc
-rw-r--r--@ 1 mason   staff       55 Jan  9 11:15 README.md
drwxr-xr-x  3 mason   staff      102 Jan  8 20:58 lib
-rw-r--r--@ 1 mason   staff      378 Jan  9 09:52 mason.gemspec
drwxr-xr-x  7 mason   staff      238 Jan  9 15:55 test-the-gem-bro
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Also note that, just to be sure, I nuked the "vendor" dir in the test-the-gem-bro project, and also the Gemfile.lock file:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ ls -l
total 16
-rw-r--r--@ 1 mason   staff   417 Jan  9 15:53 Gemfile
-rwxr-xr-x@ 1 mason   staff   131 Jan  9 09:31 test.rb
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

But it still works:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason:0x007f8c1212dc48>
Will it boogie?
w00t w00t
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Now, the $0.64 question: if we change the gem code, and do nothing else, will the test project see the changes and run the new code? Yes:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason:0x007ff1e2b347d8>
Will it boogie?
Sorry, the boogie feature has been removed in this new version.
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Sick! So, this confirms that it's hella easy to work on a gem concurrently with a project that uses it. 👍🏽
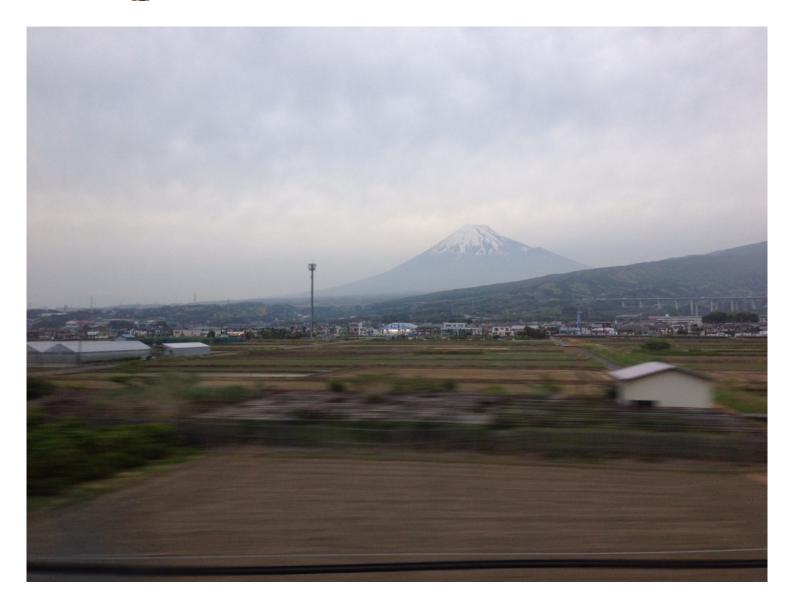
figure out how to use the gem by referencing its GitHub repo
2016-01-09

So, I moved the test project into the repo for the mason gem itself:

```
[mason@MacBook-Pro-No mason-gem (master)]$ ls -l
total 872
drwxr-xr-x  6 mason   staff      204 Jan  9 11:09 README-mason-gem
-rw-r--r--@ 1 mason   staff   431256 Jan  9 11:10 README-mason-gem.pdf
drwxr-xr-x  6 mason   staff      204 Jan  9 11:15 README-mason-gem.vpdoc
-rw-r--r--@ 1 mason   staff       55 Jan  9 11:15 README.md
drwxr-xr-x  3 mason   staff      102 Jan  8 20:58 lib
-rw-r--r--  1 mason   staff     4096 Jan  9 10:03 mason-0.0.1.gem
-rw-r--r--@ 1 mason   staff      378 Jan  9 09:52 mason.gemspec

drwxr-xr-x  5 mason   staff      170 Jan  9 10:12 test-the-gem-bro ← NEW!! 😲
[mason@MacBook-Pro-No mason-gem (master)]$
```

Then, I updated the Gemfile for the test project to reference the gem's GitHub repo, not the local path to the gem:

```
[mason@MacBook-Pro-No mason-gem (master)]$ cd test-the-gem-bro/
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ pygmentize -l ruby Gemfile

gem "mason", github: "masonmark/mason-gem"

# gem "mason", path: "../"
    # path is to parent dir, not gem itself
```

Obviously, this way does not work without a bundle install (unlike when we previously referenced the gem by its local path), because Bundler has to fetch the gem from Github:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
The git source git://github.com/masonmark/mason-gem.git is not yet checked out.
Please run `bundle install` before trying to start your application
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

That's OK though, because in a real project we always have to do this anyway:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle install --path vendor/
bundle
Fetching git://github.com/masonmark/mason-gem.git
Resolving dependencies...
Rubygems 2.0.14 is not threadsafe, so your gems will be installed one at a time.
Upgrade to Rubygems 2.1.0 or higher to enable parallel gem installation.
Using mason 0.0.1 from git://github.com/masonmark/mason-gem.git (at master@1575c6f)
Using bundler 1.11.2
Bundle complete! 1 Gemfile dependency, 2 gems now installed.
Bundled gems are installed into ./vendor/bundle.
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

And now:

```
[mason@MacBook-Pro-No test-the-gem-bro (master)]$ bundle exec ./test.rb
Here's a Mason instance: #<Mason:0x007fe0aa28c1b0>
Will it boogie?
w00t w00t
[mason@MacBook-Pro-No test-the-gem-bro (master)]$
```

Cool!