Contributor Covenant Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at me@oinak.com. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

MyGemMini

Welcome to your new gem! In this directory, you'll find the files you need to be able to package up your Ruby library into a gem. Put your Ruby code in the file `lib/my_gem_mini`. To experiment with that code, run `bin/console` for an interactive prompt.

TODO: Delete this and the text above, and describe your gem

Installation

Add this line to your application's Gemfile:

`ruby gem 'my_gem_mini'`

And then execute:

```
$ bundle
```

Or install it yourself as:

```
$ gem install my_gem_mini
```

Usage

TODO: Write usage instructions here

Development

After checking out the repo, run `bin/setup` to install dependencies. Then, run `rake test` to run the tests. You can also run `bin/console` for an interactive prompt that will allow you to experiment.

To install this gem onto your local machine, run `bundle exec rake install`. To release a new version, update the version number in `version.rb`, and then run `bundle exec rake release`, which will create a git tag for the version, push git commits and tags, and push the `.gem` file to rubygems.org.

Contributing

Bug reports and pull requests are welcome on GitHub at https://github.com/oinak/my_gem_mini. This project is intended to be a safe, welcoming space for collaboration, and contributors are expected to adhere to the [Contributor Covenant](http://contributor-covenant.org) code of conduct.

License

The gem is available as open source under the terms of the MIT License.

Code of Conduct

Everyone interacting in the MyGemMini project’s codebases, issue trackers, chat rooms and mailing lists is expected to follow the code of conduct.

Contributor Covenant Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at me@oinak.com. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at http://contributor-covenant.org/version/1/4

MyGemRspec

Welcome to your new gem! In this directory, you'll find the files you need to be able to package up your Ruby library into a gem. Put your Ruby code in the file `lib/my_gem_rspec`. To experiment with that code, run `bin/console` for an interactive prompt.

TODO: Delete this and the text above, and describe your gem

Installation

Add this line to your application's Gemfile:

`ruby gem 'my_gem_rspec'`

And then execute:

```
$ bundle
```

Or install it yourself as:

```
$ gem install my_gem_rspec
```

Usage

TODO: Write usage instructions here

Development

After checking out the repo, run `bin/setup` to install dependencies. Then, run `rake spec` to run the tests. You can also run `bin/console` for an interactive prompt that will allow you to experiment.

To install this gem onto your local machine, run `bundle exec rake install`. To release a new version, update the version number in `version.rb`, and then run `bundle exec rake release`, which will create a git tag for the version, push git commits and tags, and push the `.gem` file to rubygems.org.

Contributing

Bug reports and pull requests are welcome on GitHub at https://github.com/oinak/my_gem_rspec. This project is intended to be a safe, welcoming space for collaboration, and contributors are expected to adhere to the Contributor Covenant code of conduct.

License

The gem is available as open source under the terms of the MIT License.

Code of Conduct

Everyone interacting in the MyGemRspec project’s codebases, issue trackers, chat rooms and mailing lists is expected to follow the code of conduct.

Testing 101

madrid-rb

Grupo de usuarios de ruby de madrid

Último jueves de cada mes (en general)

http://www.madridrb.com

Yo

Fernando Martínez a.k.a. @oinak

Mantengo rubyonrails.org.es

Ayudé con conferenciaror.es

Trabajo en returnly.com

@agustincnc: «el taliban de los tests»

demoscopia

¿Quién hace tests?

¿Quién hace tests *automáticos*?

¿Quién *empieza* por tests automáticos?

¿Por qué testear?

siempre hay alguien que testea

el cliente, tras pagar

tu, tras deployar

tu, tras programar

tu, mientras programas

«**todos** tenemos
un entorno de pruebas…

algunos afortunados tienen,
un entorno *aparte* para producción»

¿Cómo testear?

clase mínima de ruby:

Almacena un nobre de usuario para saludarlo

```ruby
g = Greeter.new("Nombre")
g.say
# => Hola Nombre!
```

clase mínima de ruby:

```
Nonexistent file: ../examples/greeter.rb
```

prueba mínima:

```
$ ruby -r'./examples/greeter.rb' -e 'puts Greeter.new("Fer").say'
Hello Fer
```

¿está bien?

guardar salida para comprobar

¿cual fue el error?

¿efectos secundarios?

prueba mínima en irb/fichero:

```ruby
require './examples/greeter.rb'

greet    = Greeter.new("Fer")
expected = "Hola Fer!"
got      = greet.say

if got == expected
  puts 'ok'
else
  puts "ERROR, Expected: #{expected}, got #{got}"
end
```

```
> require './examples/greeter.rb'
=> true
> greet    = Greeter.new("Fer")
=> #<Greeter:0x000055abdd9fcd40 @name="Fer">
> expected = "Hola Fer!"
=> "Hola Fer!"
> got = greet.say
=> "Hello Fer"
> if got == expected
?>   puts 'ok'
?> else
?>   puts "ERROR, Expected: #{expected}, got #{got}"
?> end
```

```
ERROR, Expected: Hola Fer!, got Hello Fer
```

D.R.Y.

¿Cómo testear (una clase)?

test mínimo (minitest):

```ruby
require 'minitest/autorun'           # "ejecútame como test"
require 'my_obj'                     # carga lo que vas a probar

class ObjTest < Minitest::Test       # convención: TuClaseTest

  def test_what                      # ejecutará todos los test_xxxx

    o = Obj.new("Value")             # 1. prepara lo necesario

    result = o.my_method             # 2. ejecuta tu codigo

    assert_equal(result, "Expected") # 3. compara con lo esperado

    o.destroy                        # 4. dejalo como lo encontraste
  end
end
```

test mínimo (minitest):

```
Nonexistent file: ../examples/greeter_test.rb
```

prueba automática (minitest)

```
$ ruby -I./examples ./examples/greeter_test.rb
Run options: --seed 5216

# Running:

F

Finished in 0.000568s, 1759.7114 runs/s, 1759.7114 assertions/s.

  1) Failure:
GreeterTest#test_say [./examples/greeter_test.rb:8]:
Expected: "Hello Ada"
  Actual: "Hello Ada!"

1 runs, 1 assertions, 1 failures, 0 errors, 0 skips
```

test mínimo (rspec):

Primero hacemos `gem install rspec`

```
Nonexistent file: ../examples/greeter_spec.rb
```

prueba automática (rspec)

```
$ rspec -Iexamples/ examples/greeter_spec.rb
F

Failures:

  1) Greeter#say returns 'Hello Name!'
     Failure/Error: expect(greeter.say).to eq("Hello Ada!")

       expected: "Hello Ada!"
            got: "Hello Ada"

       (compared using ==)
     # ./examples/greeter_spec.rb:8:in `block (3 levels) in <top (required)>'

Finished in 0.01217 seconds (files took 0.08172 seconds to load)
1 example, 1 failure

Failed examples:

rspec ./examples/greeter_spec.rb:5 # Greeter#say returns 'Hello Name!'
```

¿Cómo testear una Gema?

```
$ bundle gem my_gem
Creating gem 'my_gem'...
Do you want to generate tests with your gem?
Type 'rspec' or 'minitest' to generate those test files now and in
the future. rspec/minitest/(none):
```

gema con minitest

```
my_gem_mini/
├── bin
│   ├── console
│   └── setup
├── CODE_OF_CONDUCT.md
├── Gemfile
├── lib
│   ├── my_gem_mini
│   │   └── version.rb
│   └── my_gem_mini.rb
├── LICENSE.txt
├── my_gem_mini.gemspec
├── Rakefile
├── README.md
└── test
    ├── my_gem_mini_test.rb
    └── test_helper.rb
```

gema con rspec

```
my_gem_rspec/
├── bin
│   ├── console
│   └── setup
├── CODE_OF_CONDUCT.md
├── Gemfile
├── lib
│   ├── my_gem_rspec
│   │   └── version.rb
│   └── my_gem_rspec.rb
├── LICENSE.txt
├── my_gem_rspec.gemspec
├── Rakefile
├── README.md
└── spec
    ├── my_gem_rspec_spec.rb
    └── spec_helper.rb
```

gemspec comun

```ruby
Gem::Specification.new do |spec|
  spec.name          = "my_gem_rspec"
  spec.version       = MyGemRspec::VERSION
  spec.authors       = ["Fernando Martínez"]
  spec.email         = ["me@oinak.com"]

  spec.summary       = %q{TODO: Write a short summary, because RubyGems requires one.}
  spec.description   = %q{TODO: Write a longer description or delete this line.}
  spec.homepage      = "TODO: Put your gem's website or public repo URL here."
  spec.license       = "MIT"

  # Prevent pushing this gem to RubyGems.org. To allow pushes either set the 'allowed_push_host'
  # to allow pushing to a single host or delete this section to allow pushing to any host.
  if spec.respond_to?(:metadata)
    spec.metadata["allowed_push_host"] = "TODO: Set to 'http://mygemserver.com'"
  else
    raise "RubyGems 2.0 or newer is required to protect against " \
      "public gem pushes."
  end
  #...
end
```

gemspec minitest

```ruby
Gem::Specification.new do |spec|
  spec.name          = "my_gem_mini"
  #...
  spec.files         = `git ls-files -z`.split("\x0").reject do |f|
    f.match(%r{^(test|spec|features)/})
  end
  spec.bindir        = "exe"
  spec.executables   = spec.files.grep(%r{^exe/}) { |f| File.basename(f) }
  spec.require_paths = ["lib"]

  spec.add_development_dependency "bundler", "~> 1.16"
  spec.add_development_dependency "rake", "~> 10.0"
  spec.add_development_dependency "minitest", "~> 5.0"
end
```

gemspec rspec

```ruby
Gem::Specification.new do |spec|
  spec.name          = "my_gem_rspec"
  #...
  spec.files         = `git ls-files -z`.split("\x0").reject do |f|
    f.match(%r{^(test|spec|features)/})
  end
  spec.bindir        = "exe"
  spec.executables   = spec.files.grep(%r{^exe/}) { |f| File.basename(f) }
  spec.require_paths = ["lib"]

  spec.add_development_dependency "bundler", "~> 1.16"
  spec.add_development_dependency "rake", "~> 10.0"
  spec.add_development_dependency "rspec", "~> 3.0"
end
```

Configurar

test_helper.rb

```
Nonexistent file: ../examples/my_gem_mini/test/test_helper.rb
```

spec_helper.rb

```
Nonexistent file: ../examples/my_gem_rspec/spec/spec_helper.rb
```

Ejecutar

Rakefile (minitest)

```
Nonexistent file: ../examples/my_gem_mini/Rakefile
```

Rakefile (rspec)

```
Nonexistent file: ../examples/my_gem_rspec/Rakefile
```

¿Qué testear (unit)?

	Incoming	**Outgoing**
Query	(1) Response	(3) Stub (test at provider)
Command	(2) State (Response)	(4) Call is made (mock/verify)

- **1:** `full_name` en una persona
- **2:** `compact!` en una colección
- **3:** `Time.now` que usaremos en un timestamp
- **4:** `Slack.notify(channel: "#alert",txt: "Error: #{msg}")`

1. incoming query

- Como lo que hemos visto
- Llamar a un método público y
- Comparar el retorno con lo esperado

2. incoming command

```ruby
class Cart
  def scan(code)
    @products << Product.new(code)
  end

  def total
    sub_total - discounts
  end

  #...
end
```

Y si no hay método `products`?

2. incoming command (cont)

Opciones:

No testear "si es privado no le importa a nadie"

Efectos "testear lo que ves"

`send(:privado)`, `get_instance_variable("@foo")`

3. outgoing query

- **este** test no debería fallar
- no ejecutar código de más
- respuesta conocida

3. outgoing query (stub)

```ruby
product = Product.new(expire_date: Date.new(2010,5,5))

def Date.today
  new(2010, 5, 6)
end

assert_equal(product.expired?, true)
```

`Date` se queda modificado

hay que *arreglarlo* a mano

3. outgoing query (stub/minitest)

```ruby
# add 'require "minitest/mock"' to test_helper.rb
class PersonTest < Minitest::Test
  def setup
    @product = Product.new(expire_date: Date.new(2010,5,5))
    @expired_date = Date.new(2010,5,6)
  end

  def test_expired_after_exired_date
    Time.stub(:now, @expired_date) do
      assert_equal(@product.expired?, true)
    end
  end
end
```

método queda restaurado al terminar el bloque

3. outgoing query (stub/spec)

```ruby
describe Product do # Rspec.describe Product do
  describe ".expired?" do
    subject{ Product.new(expire_date: Date.new(2010,5,5)) }

    context "after expire date" do
      let(:expired_date){ Date.new(2010,5,6) }
      before do
        allow(Date).to receive(:today).and_return(expired_date)
      end

      it "is expired" do
        expect(subject).to be_expired # checks with and without '?'
      end
    end
  end
end
```

3. outgoing query (stub/minispec)

```ruby
# add 'require "minitest/spec"' to test_helper.rb
describe Product do # class ProductTest < Minitest::Spec
  describe ".expired?" do
    subject{ Product.new(expire_date: Date.new(2010,5,5)) }

    describe "after expire date" do
      let(:expired_date){ Date.new(2010,5,6) }
      before do
        allow(Date).to receive(:today).and_return(expired_date)
      end

      it "is expired" do
        expect(subject).must_be :expired?
      end
    end
  end
end
```

4. outgoing command

comprobar que cambiamos el mundo

no cómo, sino qué

4. outgoing command (mock/minitest)

comprobar que cambiamos el mundo

no cómo, sino qué

clase con dependencias

```
Nonexistent file: ../examples/dep_one.rb
```