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:

2016-01-09: 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

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.