RubyGems Manuals

RubyGems User Guide

Chapters

3. Installing RubyGems

3.1 Installing RubyGems

Get it from RubyForge (http://rubyforge.org/frs/?group_id=126) and run (as root, if appropriate and necessary)

```
ruby setup.rb
```

It's easy. It installs the required library files and the **gem** command. This command gives us the power to do everything else in this document, except distribute gems (for now!).

Debian and Ubuntu: Debian and Ubuntu do not automatically include all the standard Ruby libararies in the basic Ruby package. As a result, you may need to ''apt-get'' libyaml-ruby and libzlib-ruby before you can install rubygems. Additionally, you may need to install ruby-dev in order to install gems that have C extensions. Commonly these platforms now have a "ruby-full" package that will install most of the common libraries.

3.2 Installing RubyGems in a User Directory

If a user does not have access to the standard installation location (typically `/usr/local/lib/ruby`), then they have the option of installing RubyGems in a alternate location.

Note that if you can't install RubyGems in the standard location, then you probably can't install gems in the standard gem repository location either. You need to specifiy a non-standard gem repository location via the GEM_HOME environment variable.

Use the following to install RubyGems in a user directory (here called `/home/mystuff`) with a repository named `/home/mygemrepository`):

```
$ export GEM_HOME=/home/mygemrepository
$ ruby setup.rb --prefix=/home/mystuff
```

Notes:

1. The `export` command is shell specific. Use the appropriate command for your OS and shell. For example windows users would probably say:

```
set GEM_HOME=/home/mygemrepository
```

1. Make sure you add `/home/mystuff/bin` to your path so that the `gem` command can be found.
2. Make sure you add the `GEM_HOME` setup to your profile, so that RubyGems can find the location of your gem repository.
3. If you want the gem repository to reside inside the install directory, we recommend setting `GEM_HOME` *prefix_dir*`/gems`. (where *prefix_dir* is given as the valud of `--prefix` in the config step)

3.3 Updating RubyGems

Modern Versions of RubyGems

If your RubyGems version is 0.8.5 or later, you can upgrade to the latest version with:

```
gem update --system
```

Don't forget to use `sudo` if your system requires root access to install ruby libraries.

Prior to RubyGems 0.8.5 or RubyGems 1.2.0 (or "Nothing to Update")

If your current version of RubyGems is older than version 0.8.5, or specifically RubyGems 1.2.0, or you see the message "Nothing to update" when you tried `gem update –system`, then use the following commands:

```
gem install rubygems-update
update_rubygems
```

Manual Upgrades

Download the latest RubyGems tar or zip file and following the instructions for *Installing RubyGems*.

3.4 Post-install -- Setting Up the RubyGems Environment

Now that you have RubyGems installed, you should be ready to run applications using gems, right?

Well, almost.

You have one more decision to make: How to let Ruby programs know to use the gems repository.

You see, because the of versioned nature of the gems repository, RubyGems doesn't store the library files directly in standard library search path. It adds the necessary gem packages to the library seach path as needed at run time.

This means that RubyGems must be loaded before any gem libraries are accessable.

Ruby 1.9

The default Ruby 1.9 package now includes RubyGems by default on most platforms (presently Debian based systems split this out into a separate package). This means that on Ruby 1.9 and above, you will not need to `require 'rubygems'` in order to load gem libraries.

The Hard Way

The most direct way to make RubyGems available is to just require it in the source code:

```
require 'rubygems'
require 'some_gem_library'
# ...
```

The big problem with this approach is that you don't want to make this change to *every single Ruby program you download*! While ok for quick scripts you write yourself, this is not the way to go.

Using the `-rubygems` Command Line Option

To avoid modifying all the Ruby programs you install, you could tell the `ruby` interpreter to preload ruby gems before running other software. You can easily do this by giving the `ruby` command a `-rubygems` option each time you run a program.

ruby -rubygems my_program_that_uses_gems
This works, and avoids changing installed software, but is a pain to type all the time. Fortunately there is another option.

Using `RUBYOPT`

By setting the `RUBYOPT` environment variable to the value `rubygems`, you tell Ruby to load RubyGems every time it starts up. This is similar to the `-rubygems` options above, but you only have to specify this once (rather than each time you run a Ruby script).

Unix users will want to put the following line in their `.profile` (or equivalent):

export RUBYOPT=rubygems
Windows users will want to set the RUBYOPT environment variable using the appropriate sysetm utility. (On XP you can find it under Settings / Control Panel / System. Click the advanced tab and then the “Environment Variables” button near the bottom. Note that the one-click installer will set up RUBYOPT for you automatically (unless you request it not be done).

The Future

The need to preload the RubyGems software is one of the biggest drawbacks to RubyGems’ versioned software approach. The RubyGems team is investigating ways of making this issue much less onerous.

In the meantime, enjoy RubyGems.

← previous chapter next chapter →