Джемы, зависимости, библиотеки

Андрей Васильев

2017

Наборы ПО Ruby

Наборы ПО в Ruby носят имя gem (джем), драгоценный камень

- Каталог всех публичных джемов называется rubygems.org
- Джемы предназначены для решения различных задач
  - Предоставление API для решения задач
  - Прикладные инструменты для разработчиков
  - Прикладные приложения
- Для установки и удаления используте приложение gem
- Джемы устанавливаются в каталог конкретной версией интерпретатора, не разделяются между ними
  - Джемы являются общими для проектов
  - Может быть несколько одновременно установленных версий джемов

Базовые команды инструмента gem

Поиск джемов

Для поиска по имени можно использовать

```
$ gem search png
```

Удобно использовать также и сайт rubygems.org

Установка джемов

Когда нашли подходящий джем, то его можно поставить

```
$ gem install chunky_png
```

По умолчанию установка включает в себя и документацию, для ускорения её можно отключить опцией `--no-doc`

```
$ gem install --no-doc chunky_png
```

Базовые команды инструмента gem

Отображение списка установленных джемов

Команда `list` показывает список джемов

```
$ gem list
```

Удаление джемов

Команда `uninstall` удаляет джемы

```
$ gem uninstall chunky_png
```

Если у джема были зависимости, то они были автоматически установлены, но не будут автоматически удалены

Просмотр документации

Если во время установки джема вы поставили документацию, тогда её можно посмотреть с помощью встроенных средств

Инструмент `ri` позволяет просмотреть документацию

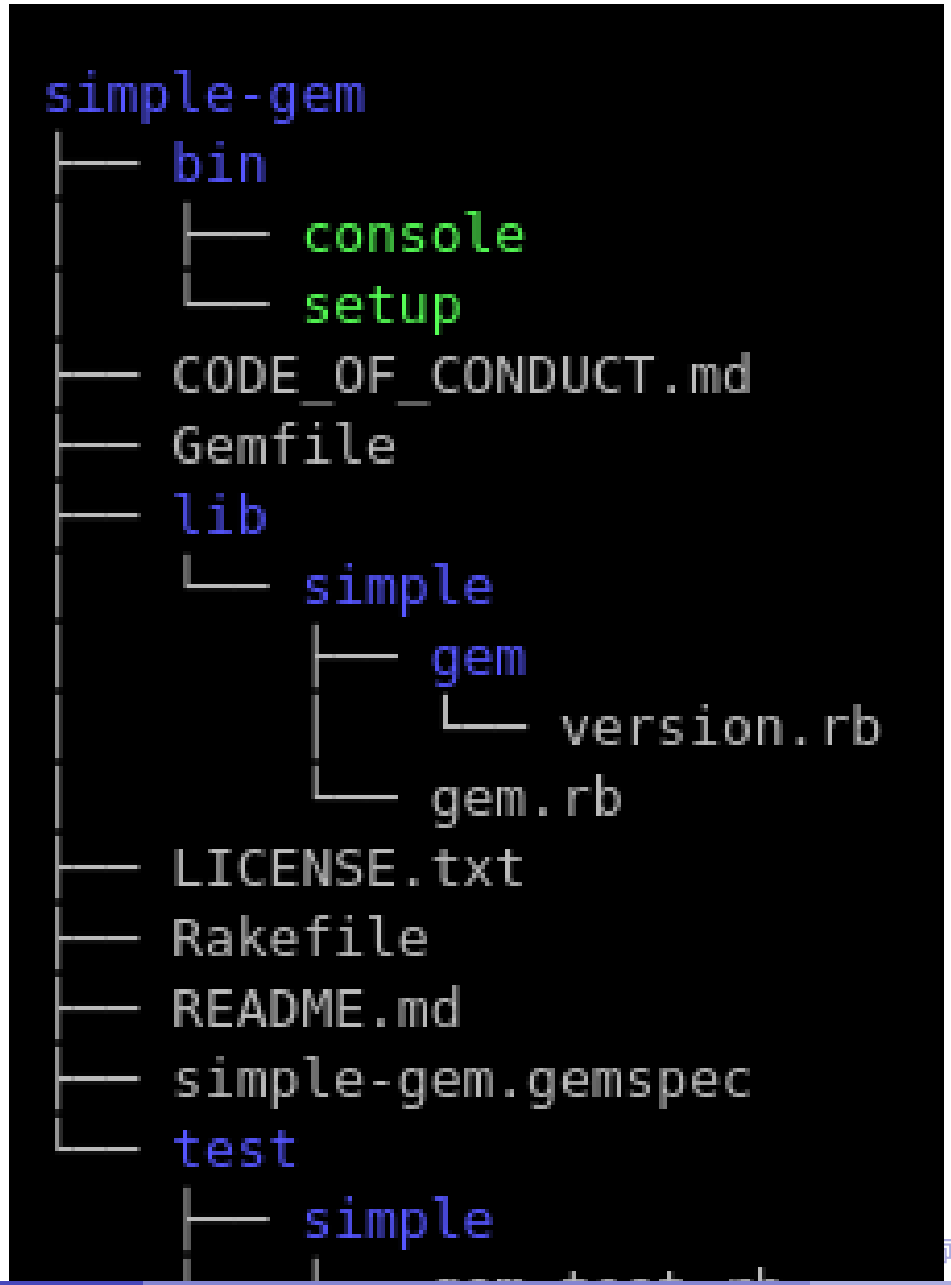
```
$ ri chunky_png
```

Вы можете запустить встроенный сервер для просмотра документации

```
$ gem server
```

Большинство джемов также предоставляют отдельные сайты с детальной документацией, ссылки на них можно найти на сайте rubygems.org

Структура джема



Структура джема, расшифровка

- `bin` - исполняемые файлы джема
- `lib` - библиотечные файлы джема
- `test` - каталог с автоматическими тестами
- `Gemfile` - список зависимостей джема
- `Rakefile` - конфигурация автоматических задач
- `README.md` - краткое описание джема для людей
- `.gemspec` - спецификация джема

Ключевые свойства спецификации

- Название, например `simple-gem`
- Версия, например `1.0.5`
- Описание краткое и полное

Проблемы совместной разработки

- Как вместе редактировать общий исходный код?
- Какие версии джемов использованы для работы приложения?
- Как следует оформлять исходный код приложения?
- Какие типы автоматических тестов надо написать в приложении?
- Как должен быть устроен процесс разработки?

Джем Bundler

Джем bundler является стандартом де-факто в Ruby сообществе для управления зависимостями

- Указывает список джемов, использующихся при разработке, тестирования и поставке приложения
- Позволяет запускать приложение в окружении джемов, которые указаны в списке, что изолирует от других версий
- Позволяет указывать конкретные версии джемов
- Создаёт шаблон джема для быстрого старта

Для начала использования bundler, его необходимо установить:

```
gem install bundler
```

Список зависимостей, Gemfile

Набор необходимых джемов в Bundler указывается в `Gemfile`

```
source 'https://rubygems.org'
gem 'nokogiri'
gem 'rack', '~> 2.0.1'
gem 'rspec'
```

- `source` указывает источник для получения джем-файлов
  - Их может быть несколько в одном файле
  - Сайт `rubygems.org` - источник общественнх джемов
- `gem` указывает джем, который надо установить
  - Первый аргумент - название джема
  - Последующие указывают версии джемов

Установка зависимостей

Процесс установки зависимостей очень прост:

1. Установить джем bundler: `gem install bundler`
2. Создать файл Gemfile в каталоге проекта. Можно воспользоваться командой: `bundle init`
3. Указать в файле источник для скачивания джемов
4. Указать в файле список джемов-зависимостей
5. Выполнить команду `bundle install`

Установка зависимостей на других компьютерах

После выполнения установки был создан `Gemfile.lock`, его тоже **необходимо** переносить на другие компьютеры - там указаны конкретные версии джемов, а не пожелания.
На других компьютерах выполняем `bundle install`

Установка зависимостей. Продолжение.

Добавление новых джемов в набор

Процесс добавления достаточно прост:

1. Добавить новую запись в Gemfile
2. Выполнить команду `bundle install` для установки

Использование команды add

Если у вас уже есть Gemfile, тогда можно воспользоваться командой `bundle add` для устаовки джемов:

```
bundler add write_xlsx
```

Команда добавит строку в Gemfile и установит джем

Получение информации о функциях Bundler

Bundler предоставляет обширную документацию по встроенным командам

- Для отображения списка команд выполните `bundle help`
- Для получения помощи по конкретной команде выполните `bundle help <command>`, вместо `<command>` надо написать название команды, например: `bundle help add`

Интересные команды

- `bundle console` - запустить IRB-сессию с установленными джемами
- `bundle clean` - удалить неиспользуемые версии джемов
- `bundle config` - настроить Bundler

Обновление версий джемов

Обычно в Gemfile указаны пожелания, а в Gemfile.lock конкретные версии приложений. Бывает необходимо обновить версию джема ввиду выхода новой версии

- с исправлениями ошибок, в том числе безопасности;
- с добавлением новой, необходимой функциональности.

Для решения этой задачи необходимо

1. Исправьте строки в Gemfile, укажите нужные версии.
2. Выполните команду `bundle update`. Команда обновит версии джемов в Gemfile.lock и установит их

Можно опустить первый шаг, однако такое обновление будет неконтролируемым - вы можете поставить слишком новые версии

Запуск приложений в рамках набора джемов

Команда `bundle exec <command>` позволяет запустить приложение в контексте установленного набора джемов

Для запуска приложения в рамках набора используйте

```
bundle exec ruby bin/application.rb
```

В таком случае в приложении `application.rb` можно подключать только джемы из набора Bundler

Также можно запускать исполняемые файлы джемов:

```
bundle exec rubocop lib
bundle exec rspec spec/my_spec.rb
```

Лучшие практики по применению Ruby

Ruby является очень выразительным языком, позволяющим решить задачу многими способами. За время жизни языка некоторые из способов были признаны сложными для восприятия и не рекомендуются к использованию

Для поддержания хорошего стиля кодирования на Ruby был разработан Ruby Style Guide
https://github.com/bbatsov/ruby-style-guide

Который также был переведён на русский язык
https://github.com/arbox/ruby-style-guide/blob/master/README-ruRU.md

Джем Rubocop

Джем Rubocop проверяет исходный код на соответствие Ruby Style Guide, а также внутренним требованиям команды

Установку джема в Bunler рекомендуется выполнять так:

```
gem 'rubocop', require: false
```

Проверка исходного кода

Для проверки всех файлов в текущем каталоге достаточно вызывать исполняемый файл джема:

```
$ bundle exec rubocop
```

Для проверки конкретных файлов и директорий:

```
$ bundle exec rubocop lib bin/application.rb
```

Автоматическое исправление расхождений

Поначалу прочитать гид и следовать ему бывает сложно. Для решения простейших проблем джем Rubocop предоставляет режим автоматического испправления:

```
$ bundle exec rubocop -a
```

Для ускорения проверки, её можно запустить в несколько потоков с помощью ключа `-p`, `--parallel`:

```
bundle exec rubocop -p
```