Комфортное управление пакетами в Ruby

backend 12 сентября 2016

RVM — Ruby Version Manager (менеджер управления пакетами в ruby) сослужит хорошую службу в тех частых ситуациях, когда вам может потребоваться запуск нескольких версий ruby на одной машине или же нескольких проектов с разными наборами gem'ов для них. Чем-то напоминает virtualenv, если сравнивать с аналогичным приложением для python.

Данный пост предположительно рассчитан на новоиспечённых рубистов и предназначен для облегчения и в какой-то мере даже автоматизации их труда.

Что такое gem?

Gem (драгоценный камень, жемчужина) — пакет ruby, предоставляющий какую-то функциональность. Например, библиотека `redcarpet`, доступная в виде gem'а, обеспечивает поддержку markdown.

Что такое gemset?

Gemset (set: набор, комплект) — набор gem'ов.

Установка RVM

```
$ \curl -L https://get.rvm.io | bash
Downloading RVM branch master
% Total    % Received % Xferd  Average Speed
                               Dload  Upload
100   124  100   124    0      0     107
100 1084k  100 1084k    0      0   234k
```

```
* To start using RVM you need to run
`source /home/user/.rvm/scripts/rvm`
```

Перезапускаем консоль и убеждаемся в том, что `rvm` был успешно установлен:

```
$ rvm -v
rvm 1.23.13

$ rvm get stable --auto-dotfiles
```

Дополним переменную PATH, дабы указать путь к директории rvm. Запись нужно сделать в `.bashrc` или `.zshenv`.

```
# .bashrc
export PATH="$PATH:$HOME/.rvm/bin"
```

Дальше любопытно было бы просмотреть список доступных для установки и уже установленных пакетов ruby.

```
$ rvm list known # просмотреть все доступные для установки версии Ruby
$ rvm list       # все установленные версии Ruby
=* ruby-1.9.3-p448 [ x86_64 ]
# => - current - текущая версия
# =* - current && default - текущая и спользуется по-умолчанию
#  * - default - используемая по-умолчанию версия
```

Использование разных версий ruby и gemset'ы

```
$ source ~/.rvm/scripts/rvm # активировать rvm
$ rvm install 1.9.3 # установить несколько версий ruby
$ rvm install 2.0.0
$ rvm use ruby-2.0.0-p247 --default # переключиться на ruby 2 и использовать v2 по-умол
$ rvm list
   ruby-1.9.3-p448 [ x86_64 ]
=* ruby-2.0.0-p247 [ x86_64 ]

# => - current
# =* - current && default
#  * - default
```

Для примера создадим gemset `jekyll`, где будут собраны все нужные одноимённому генератору статических страниц gem'ы.

```
$ source ~/.rvm/scripts/rvm # «включаем» rvm
$ rvm use 1.9.3@jekyll --create # создать gemset
$ rvm use 1.9.3@jekyll --default # выбирать его по-умолчанию
$ gem install jekyll jekyll-tagging i18n # установка gem'ов
$ gem list # просмотр установленных в выбранном gemset'е gem'ов
i18n (0.6.5)
jekyll (1.2.1)
jekyll-tagging (0.5.0)
liquid (2.5.3)
maruku (0.7.0)
```

Gemset'ы создаются для определённой версии ruby. Так, при переключении с одной версии на другую вы будете всегда видеть разный набор gemset'ов: отображаются они также лишь для своей версии.

```
$ rvm gemset list # набор gemset'ов для ruby2

gemsets for ruby-2.0.0-p247
   (default)
=> example
   global

$ cd workspace/jekyll # переход в директорию проекта, где используется ruby1.9
$ rvm gemset list     # набор gemset'ов для ruby1.9

gemsets for ruby-1.9.3-p448
   (default)
   global
=> jekyll
```

Но здесь мы забежали чуть вперёд. Дело в том, что при переходе в директорию проекта, автоматически меняется версия ruby и набор gemset'ов, о чём речь пойдёт чуть ниже.

Gemset'ы можно удалять(delete), очищать(), экспортировать(export) и импортировать(import) gem'ы из одного в gemset'а в другой. Наиболее часто используемые действия отражены в данной ниже таблице.

команда	описание
rvm gemset create	создать новый gemset
rvm gemset export new.gems	экспорт списка gem'ов в файл new.gems
rvm gemset import new.gems	импорт в текущий gemset списка gem'ов из файла new.gems
rvm gemset delete	удалить gemset
rvm gemset empty	очистить gemset
rvm gemset delete example --force	полностью удалить gemset example
rvmreset	перезагрузка RVM
rvm uninstall	удалить версию Ruby, оставить исходники
rvm implode	полностью удалить RVM!
rvm list known	получить список всех версий ruby доступных для установки
rvm list	отобразить список установленных версий ruby
rvm gemset list	просмотреть список gemset'ов в выбранной версии ruby
rvm install 1.9.3	установить ruby версии 1.9.3
rvm remove 1.9.3	удалить ruby версии 1.9.3
rvm use 1.9.3	переключиться на ruby версии 1.9.3
rvm use 1.9.3@jekyll --default	использовать версию ruby 1.9.3 с gemset jekyll по умолчанию
rvm use system	использовать системную версию ruby

Создание отдельного окружения для проекта

Бывают случаи, когда необходимо не просто использовать разные версии ruby, но и разные версии gemset'ов для разных проектов. В этом посильную помощь может оказать создание своего окружения для каждого из проектов.

В примере выше мы создали gemset `jekyll`, который использует версию ruby 1.9.3 и собственный набор gem'ов. Теперь у нас имеется два пути для работы с ним.

Переключение между gemset'ами

Во-первых, можно переключаться между gemsets при необходимости.

```
$ rvm gemset use global
$ rvm gemset use jekyll
```

Окружение?

Но зачастую быстрее один раз внести необходимые настройки. Для этого следует создать два файла в каталоге с проектом: `.ruby-gemset` и `.ruby-version`, куда и поместить информацию об используемом в проекте gemset'е и версии ruby соответственно.

```
$ cat ~/workspace/jekyll/.ruby-gemset
jekyll

$ cat ~/workspace/jekyll/.ruby-version
ruby-1.9.3-p448
```

Теперь достаточно просто зайти в директорию проекта, и нужные нам параметры будут установлены автоматически.

```
$ cd workspace/jekyll
$ rvm list

=> ruby-1.9.3-p448 [ x86_64 ]
 * ruby-2.0.0-p247 [ x86_64 ]

# => - current
```

```
# =* - current && default
#  * - default
```

То есть, если вы использовали версию ruby 2, после перехода в каталог с проектом, она автоматически изменится на версию, указанную в `.ruby-version`, что достаточно удобно.

Покорение Jekyll Ключи от всех дверей

КОММЕНТАРИИ