LEARNING ruby-on-rails-4

#ruby-on-rails-4

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ruby-on-rails-4

It is an unofficial and free ruby-on-rails-4 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ruby-on-rails-4.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ruby-on-rails-4

Remarks

This section provides an overview of what ruby-on-rails-4 is, and why a developer might want to use it.

It should also mention any large subjects within ruby-on-rails-4, and link out to the related topics. Since the Documentation for ruby-on-rails-4 is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Setup Ruby On Rails on Ubuntu 16.04 Xenial Xerus

All commands should be run in Linux terminal (hotkey: `Ctrl` + `Alt` + `T`)

You need to install Ruby on your local machine in development environment. The first step is to install some dependencies for Ruby.

```
sudo apt-get update
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev
libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev python-
software-properties libffi-dev
```

You can install Ruby using one of three methods. Some of these conflict with each other, so choose the one that sounds the most interesting to you.

First method: installing with `rbenv`

```
cd
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
exec $SHELL

git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL

rbenv install 2.3.3
rbenv global 2.3.3
ruby -v
```

The last step is to install Bundler:

```
gem install bundler
```

Second method: Installation with `rvm`

```
sudo apt-get install libgdbm-dev libncurses5-dev automake libtool bison libffi-dev
gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
curl -sSL https://get.rvm.io | bash -s stable
source ~/.rvm/scripts/rvm
rvm install 2.3.3
rvm use 2.3.3 --default
ruby -v
```

The last step is to install Bundler:

```
gem install bundler
```

The third method: Installing from source

```
cd
wget http://ftp.ruby-lang.org/pub/ruby/2.3/ruby-2.3.3.tar.gz
tar -xzvf ruby-2.3.3.tar.gz
cd ruby-2.3.3/
./configure
make
sudo make install
ruby -v
```

The last step is to install Bundler:

```
gem install bundler
```

Installing Rails

Rails ships with many dependencies these days, we're going to need to install a JavaScript run time like NodeJS. This lets you use Coffee script and the Asset Pipeline in Rails which combines and minifies your JavaScript to provide a faster production environment.

```
curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Now Install `Rails 4.2.6`

```
gem install rails -v 4.2.6 (you may set a specific version of rails)
```

If you are using `rbenv` run the following command:

```
rbenv rehash
```

Then run `rails -v` to make sure you have everything installed correctly. It should get you your rails version. `# Rails 4.2.6`

Setup Ruby On Rails on Ubuntu 16.10 Yakkety Yak

This will take about 30 minutes. We will be setting Ruby on Rails Development Environment on Ubuntu 16.10 Yakkety Yak.

Open up your terminal using `Ctrl` + `Alt` + `T`.

Installing Ruby

The First step is to install Dependencies For Ruby.

```
sudo apt-get update
sudo apt-get install git-core curl zlib1g-dev build-essential libssl-dev libreadline-dev libyaml-dev libsqlite3-dev sqlite3 libxml2-dev libxslt1-dev libcurl4-openssl-dev python-software-properties libffi-dev nodejs
```

We will be using Ruby version 2.4.0. Install Ruby using one of the three methods. Each have their own benefits, mostly people prefer rbenv but if you are familiar with rvm then go for it.

Note: Only `rbenv` users need to run `rbenv rehash` after installing `bundler`.

Installing with `rbenv`:

```
cd
git clone https://github.com/rbenv/rbenv.git ~/.rbenv
echo 'export PATH="$HOME/.rbenv/bin:$PATH"' >> ~/.bashrc
echo 'eval "$(rbenv init -)"' >> ~/.bashrc
exec $SHELL

git clone https://github.com/rbenv/ruby-build.git ~/.rbenv/plugins/ruby-build
echo 'export PATH="$HOME/.rbenv/plugins/ruby-build/bin:$PATH"' >> ~/.bashrc
exec $SHELL

rbenv install 2.4.0
rbenv global 2.4.0
ruby -v
```

Then install Bundler:

```
gem install bundler
```

Use `rbenv rehash` after installing bundler.

Installing with `rvm`:

```
sudo apt-get install libgdbm-dev libncurses5-dev automake libtool bison libffi-dev
gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E3
curl -sSL https://get.rvm.io | bash -s stable
```

```
source ~/.rvm/scripts/rvm
rvm install 2.4.0
rvm use 2.4.0 --default
ruby -v
```

Then install Bundler:

```
gem install bundler
```

Installing with source:

```
cd
wget http://ftp.ruby-lang.org/pub/ruby/2.4/ruby-2.4.0.tar.gz
tar -xzvf ruby-2.4.0.tar.gz
cd ruby-2.4.0/
./configure
make
sudo make install
ruby -v
```

Then Last step is to install Bundler:

```
gem install bundler
```

Installing Rails

Rails ships with so many dependencies these days, we're going to need to install a JavaScript run-time like NodeJS. This lets you use Coffee-script and the Asset Pipeline in Rails which combines and minifies your JavaScript to provide a faster production environment.

Install NodeJS using the official repository:

```
curl -sL https://deb.nodesource.com/setup_4.x | sudo -E bash -
sudo apt-get install -y nodejs
```

Now install `Rails 4.2.7`:

```
gem install rails -v 4.2.7.1
```

If you are using `rbenv` then run the following command to make rails executable available:

```
 rbenv rehash
```

Now Rails is Installed, Run `rails -v` to make sure rails installed properly:

```
rails -v
# Rails 4.2.7.1
```

If you are getting a different result for some reason, it means that your environment may not be setup properly.

Setting Up PostgreSQL

Rails ships with sqlite3 as the default database. Chances are you won't want to use it because it's stored as a simple file on disk. You'll probably want something more robust like MySQL or PostgreSQL.

There is a lot of documentation on both, so you can just pick one that seems like you'll be more comfortable with.

```
sudo sh -c "echo 'deb http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main' >
/etc/apt/sources.list.d/pgdg.list"
wget --quiet -O - http://apt.postgresql.org/pub/repos/apt/ACCC4CF8.asc | sudo apt-key add -
sudo apt-get update
sudo apt-get install postgresql-common
sudo apt-get install postgresql-9.5 libpq-dev
```

PostgreSQL installation doesn't setup a user for you, so you'll need to follow these steps to create a user with permission to create databases. Feel free to replace `Hizqeel` with your username.

```
sudo -u postgres createuser hizqeel -s

# If you would like to set a password for the user, you can do the following
sudo -u postgres psql
postgres=# \password hizqeel
```

Read Getting started with ruby-on-rails-4 online: https://riptutorial.com/ruby-on-rails-4/topic/4014/getting-started-with-ruby-on-rails-4

Chapter 2: Basic of Strong Parameters

Introduction

Examples

What is strong parameters?

As edgeapi suggests, it provides an interface for protecting attributes from end-user assignment. This makes Action Controller parameters forbidden to be used in Active Model mass assignment until they have been whitelisted.

In addition, parameters can be marked as required and flow through a predefined raise/rescue flow to end up as a 400 Bad Request with no effort. Please review the `PeopleController` file and the comments listed to see how to use strong parameters?

```
class PeopleController < ActionController::Base
  # Using "Person.create(params[:person])" would raise an
  # ActiveModel::ForbiddenAttributesError exception because it'd
  # be using mass assignment without an explicit permit step.
  # This is the recommended form:
  def create
    Person.create(person_params)
  end

  # This will pass with flying colors as long as there's a person key in the
  # parameters, otherwise it'll raise an ActionController::MissingParameter
  # exception, which will get caught by ActionController::Base and turned
  # into a 400 Bad Request reply.
  def update
    redirect_to current_account.people.find(params[:id]).tap { |person|
      person.update!(person_params)
    }
  end

  private
    # Using a private method to encapsulate the permissible parameters is
    # just a good pattern since you'll be able to reuse the same permit
    # list between create and update. Also, you can specialize this method
    # with per-user checking of permissible attributes.
    def person_params
      params.require(:person).permit(:name, :age)
    end
end
```

Read Basic of Strong Parameters online: https://riptutorial.com/ruby-on-rails-4/topic/9078/basic-of-strong-parameters

Credits

S. No	Chapters	Contributors
1	Getting started with ruby-on-rails-4	Community, Hizqeel, Kirill Zhuravlov
2	Basic of Strong Parameters	TheMouseMan