

Rails Environment & Configuration

Sang Shin

JPassion.com

“Code with Passion!”



Topics

- Gem and RubyGem
- Gemfile and Gemfile.lock
- Bundler
- Environments
- Configuration
- Initializers
- Rake

Gem & RubyGem (per system)

What is Gem and RubyGem?

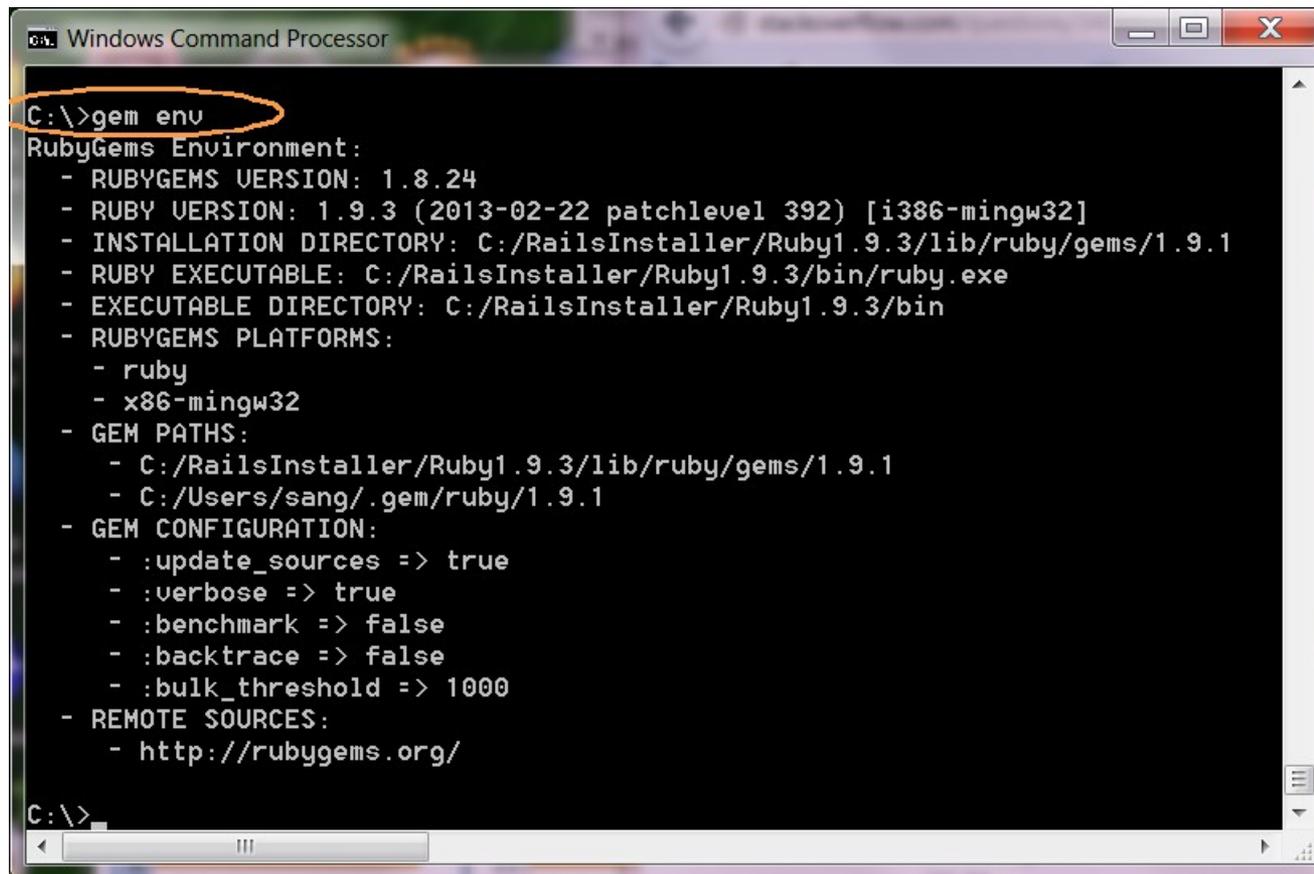
- Gem is a standard package format for Ruby libraries
 - > In the same way “jar” and “war” are standard package formats for Java libraries
- In Ruby/Rails world, a feature is built as a gem and there are tons of 3rd-party gems you can use right away
 - > Security, testing, performance, ...
- You can create your own custom gems or extend existing gems
- RubyGem is package manager for the Ruby programming language
 - > Manage the installation/uninstallation of gems (per your system)
 - > RubyGem comes with Ruby 1.9+
 - > You interact with RubyGem with “gem” command

What about Rails?

- Rails itself is a gem
 - > With dependencies on many other gems
- Special gems – gems that provide commands that can be executed at the command line
 - > rails
 - > bundler
 - > rake

Gem Environment

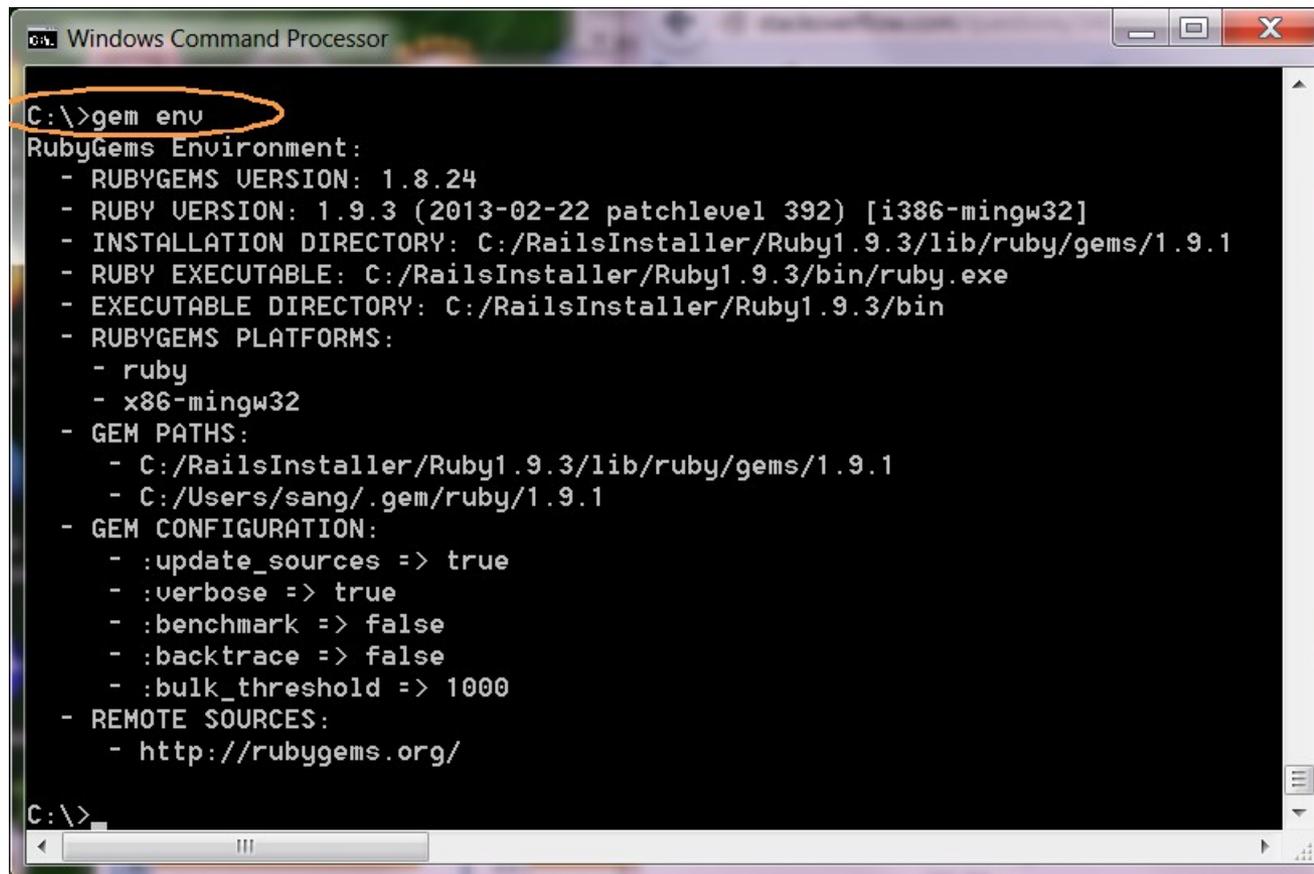
- “gem env”



```
C:\>gem env
RubyGems Environment:
- RUBYGEMS VERSION: 1.8.24
- RUBY VERSION: 1.9.3 (2013-02-22 patchlevel 392) [i386-mingw32]
- INSTALLATION DIRECTORY: C:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1
- RUBY EXECUTABLE: C:/RailsInstaller/Ruby1.9.3/bin/ruby.exe
- EXECUTABLE DIRECTORY: C:/RailsInstaller/Ruby1.9.3/bin
- RUBYGEMS PLATFORMS:
  - ruby
  - x86-mingw32
- GEM PATHS:
  - C:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1
  - C:/Users/sang/.gem/ruby/1.9.1
- GEM CONFIGURATION:
  - :update_sources => true
  - :verbose => true
  - :benchmark => false
  - :backtrace => false
  - :bulk_threshold => 1000
- REMOTE SOURCES:
  - http://rubygems.org/
C:\>
```

Gem Environment

- “gem env”



```
C:\>gem env
RubyGems Environment:
- RUBYGEMS VERSION: 1.8.24
- RUBY VERSION: 1.9.3 (2013-02-22 patchlevel 392) [i386-mingw32]
- INSTALLATION DIRECTORY: C:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1
- RUBY EXECUTABLE: C:/RailsInstaller/Ruby1.9.3/bin/ruby.exe
- EXECUTABLE DIRECTORY: C:/RailsInstaller/Ruby1.9.3/bin
- RUBYGEMS PLATFORMS:
  - ruby
  - x86-mingw32
- GEM PATHS:
  - C:/RailsInstaller/Ruby1.9.3/lib/ruby/gems/1.9.1
  - C:/Users/sang/.gem/ruby/1.9.1
- GEM CONFIGURATION:
  - :update_sources => true
  - :verbose => true
  - :benchmark => false
  - :backtrace => false
  - :bulk_threshold => 1000
- REMOTE SOURCES:
  - http://rubygems.org/
C:\>
```

“gem” commands (1)

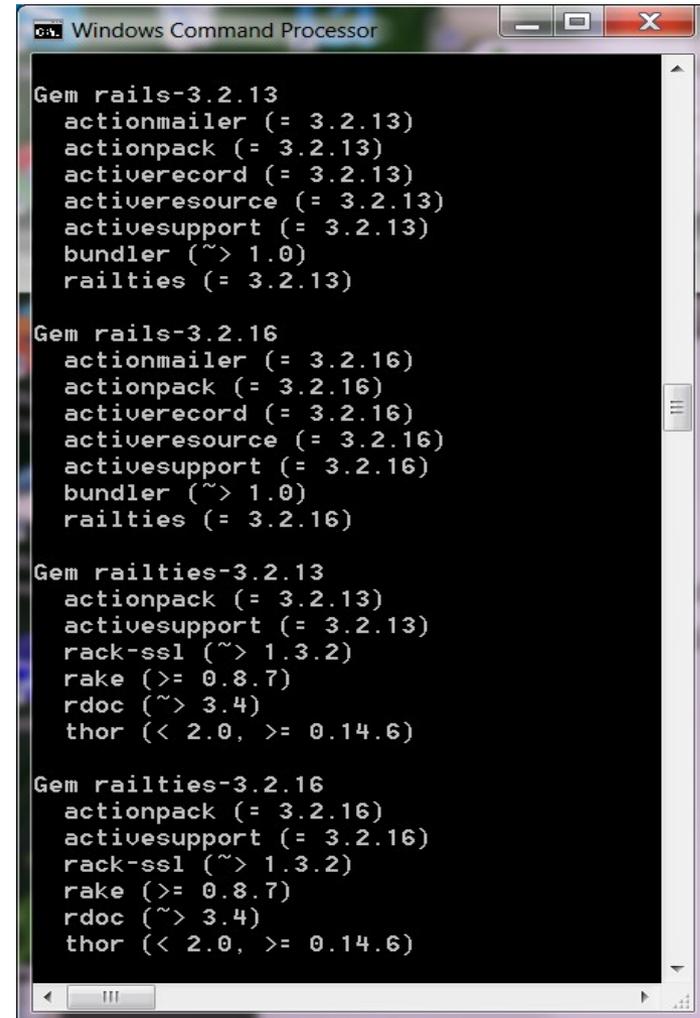
- “gem env”
 - > Display gem environment information
- “gem install <gem-name>”
 - > Installs <gem-name> gem
 - > In general, for a particular Rails app, you will want to use “bundle install” with “Gemfile” to install all required gems in a single command instead of installing each gem individually using “gem install ..” command
- “gem uninstall <gem-name>”
 - > Uninstalls <gem-name> gem

“gem” commands (2)

- “gem list --local”
 - > List gems that are previously downloaded
- “gem list --remote”
 - > List available gems from “rubygems.org” website
- “gem search <gem-name> --remote”
 - > Search for <gem-name>
- “gem dependency”
 - > Show you all gems with their dependencies
- “gem dependency -R”
 - > Show you all gems with their dependencies
 - > Also shows gems with reverse dependencies (which gems have dependency on the gems)

“rails” gem & its dependencies

- “gem dependency”



```
Windows Command Processor

Gem rails-3.2.13
actionmailer (= 3.2.13)
actionpack (= 3.2.13)
activerecord (= 3.2.13)
activeresource (= 3.2.13)
activesupport (= 3.2.13)
bundler (~> 1.0)
railties (= 3.2.13)

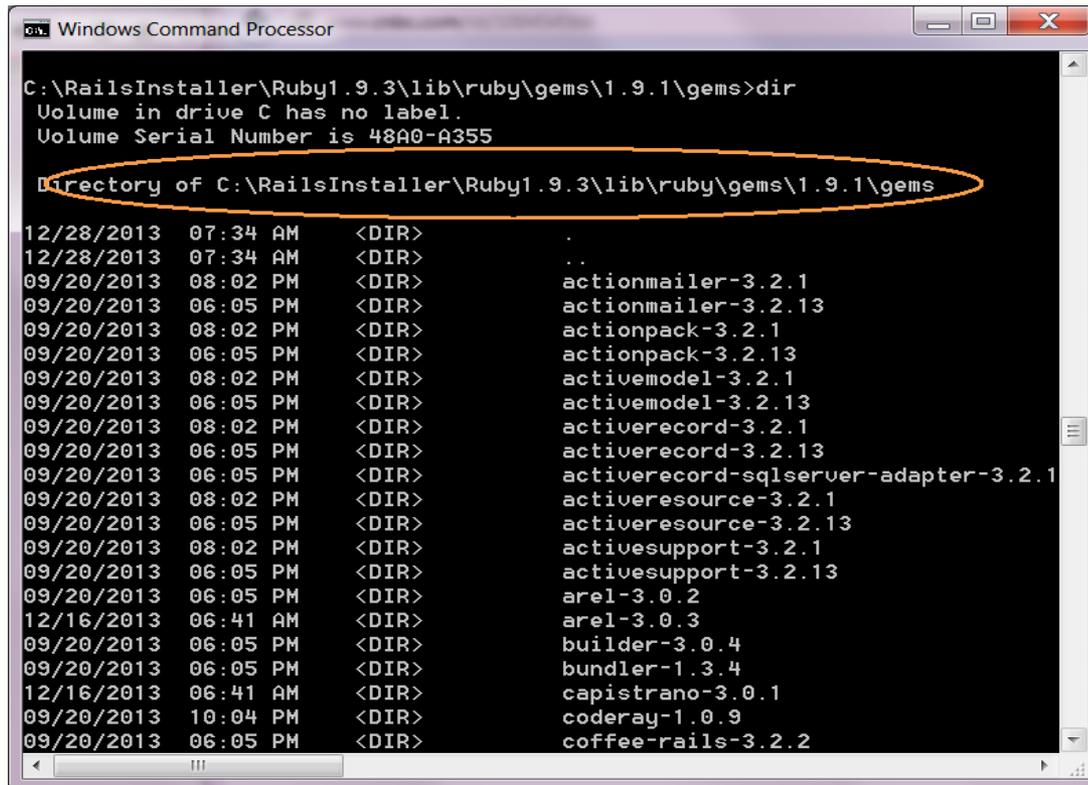
Gem rails-3.2.16
actionmailer (= 3.2.16)
actionpack (= 3.2.16)
activerecord (= 3.2.16)
activeresource (= 3.2.16)
activesupport (= 3.2.16)
bundler (~> 1.0)
railties (= 3.2.16)

Gem railties-3.2.13
actionpack (= 3.2.13)
activesupport (= 3.2.13)
rack-ssl (~> 1.3.2)
rake (>= 0.8.7)
rdoc (~> 3.4)
thor (< 2.0, >= 0.14.6)

Gem railties-3.2.16
actionpack (= 3.2.16)
activesupport (= 3.2.16)
rack-ssl (~> 1.3.2)
rake (>= 0.8.7)
rdoc (~> 3.4)
thor (< 2.0, >= 0.14.6)
```

Where Gems are stored?

- <Ruby-Home>/lib/ruby/gems/<version>/gems
- <User-Home>/gem/ruby/1.9.1



```
Windows Command Processor
C:\RailsInstaller\Ruby1.9.3\lib\ruby\gems\1.9.1\gems>dir
Volume in drive C has no label.
Volume Serial Number is 48A0-A355

Directory of C:\RailsInstaller\Ruby1.9.3\lib\ruby\gems\1.9.1\gems
12/28/2013 07:34 AM <DIR> .
12/28/2013 07:34 AM <DIR> ..
09/20/2013 08:02 PM <DIR> actionmailer-3.2.1
09/20/2013 06:05 PM <DIR> actionmailer-3.2.13
09/20/2013 08:02 PM <DIR> actionpack-3.2.1
09/20/2013 06:05 PM <DIR> actionpack-3.2.13
09/20/2013 08:02 PM <DIR> activemodel-3.2.1
09/20/2013 06:05 PM <DIR> activemodel-3.2.13
09/20/2013 08:02 PM <DIR> activerecord-3.2.1
09/20/2013 06:05 PM <DIR> activerecord-3.2.13
09/20/2013 06:05 PM <DIR> activerecord-sqlserver-adapter-3.2.1
09/20/2013 08:02 PM <DIR> activeresource-3.2.1
09/20/2013 06:05 PM <DIR> activeresource-3.2.13
09/20/2013 08:02 PM <DIR> activesupport-3.2.1
09/20/2013 06:05 PM <DIR> activesupport-3.2.13
09/20/2013 06:05 PM <DIR> arel-3.0.2
12/16/2013 06:41 AM <DIR> arel-3.0.3
09/20/2013 06:05 PM <DIR> builder-3.0.4
09/20/2013 06:05 PM <DIR> bundler-1.3.4
12/16/2013 06:41 AM <DIR> capistrano-3.0.1
09/20/2013 10:04 PM <DIR> coderay-1.0.9
09/20/2013 06:05 PM <DIR> coffee-rails-3.2.2
```

3rd Party Gems

- <http://rubygems.org/>
 - > Default Ruby gems host (4300 gems hosted Jan. 2014)
- <https://www.ruby-toolbox.com/>
 - > Show available gems based on categories
 - > Gems in the same category can be listed in the order of their popularity

Lab:

**Exercise 1: Gem & RubyGem
5524_rails4_configuration.zip**

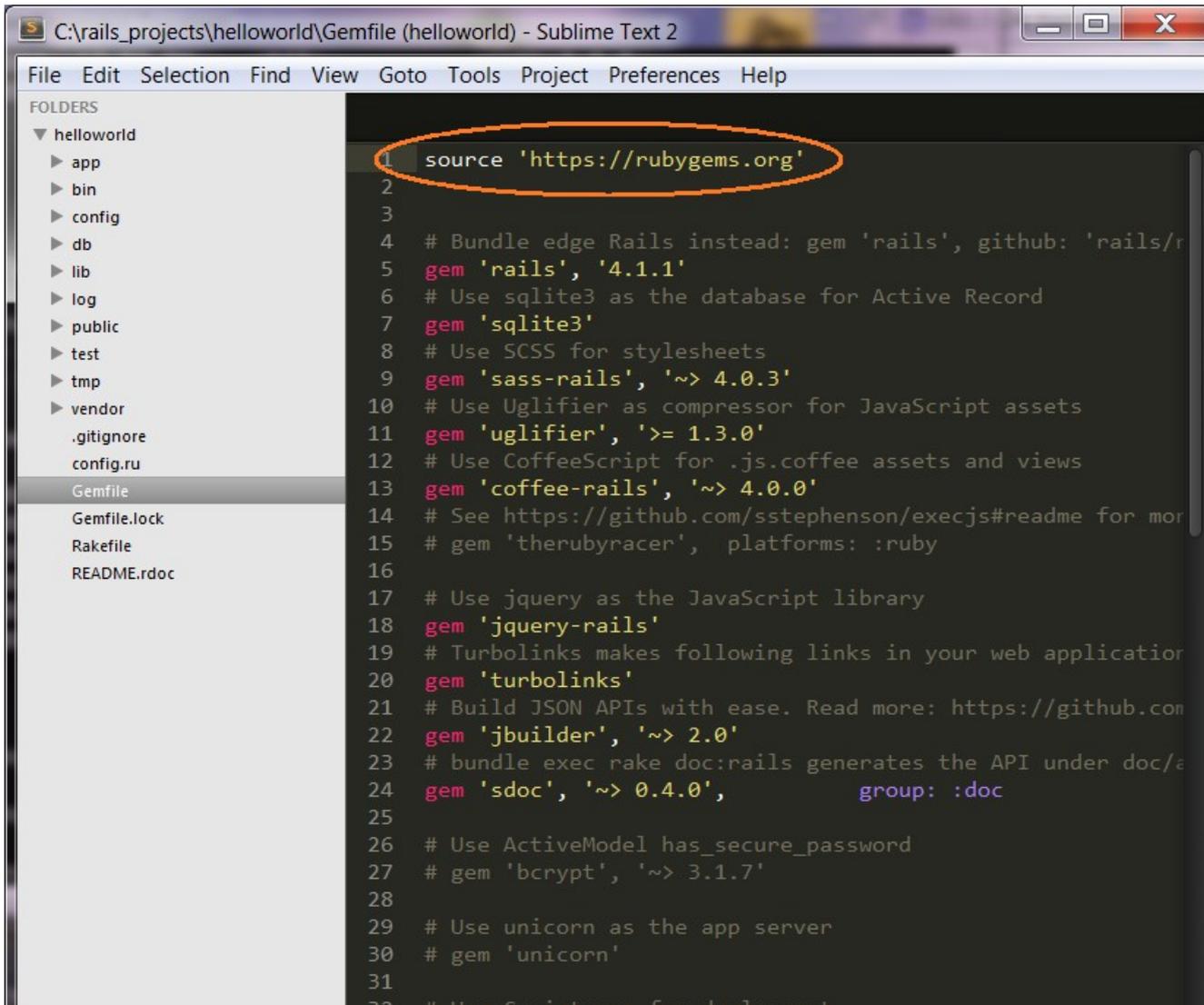


Gemfile & Gemfile.lock (per each Rails application)

What is “Gemfile”?

- Specifies all the gems (in other words, dependencies) required for a Rails app
- When you create a new Rails app through “rails new <appname>”, Rails creates a *Gemfile* for you
- You can then add, remove, update, and group gems by modifying *Gemfile*
 - > For example, if you need a security gem, you add it to the Gemfile
- Every time a modification is made to the *Gemfile*, “bundle install” or “bundle update” needs to be executed
 - > “bundle install” or “bundle update” will download and install a newly added gem, for example
 - > At the end of “rails new <new-app>”, “bundle install” gets executed by default

“Gemfile” Example



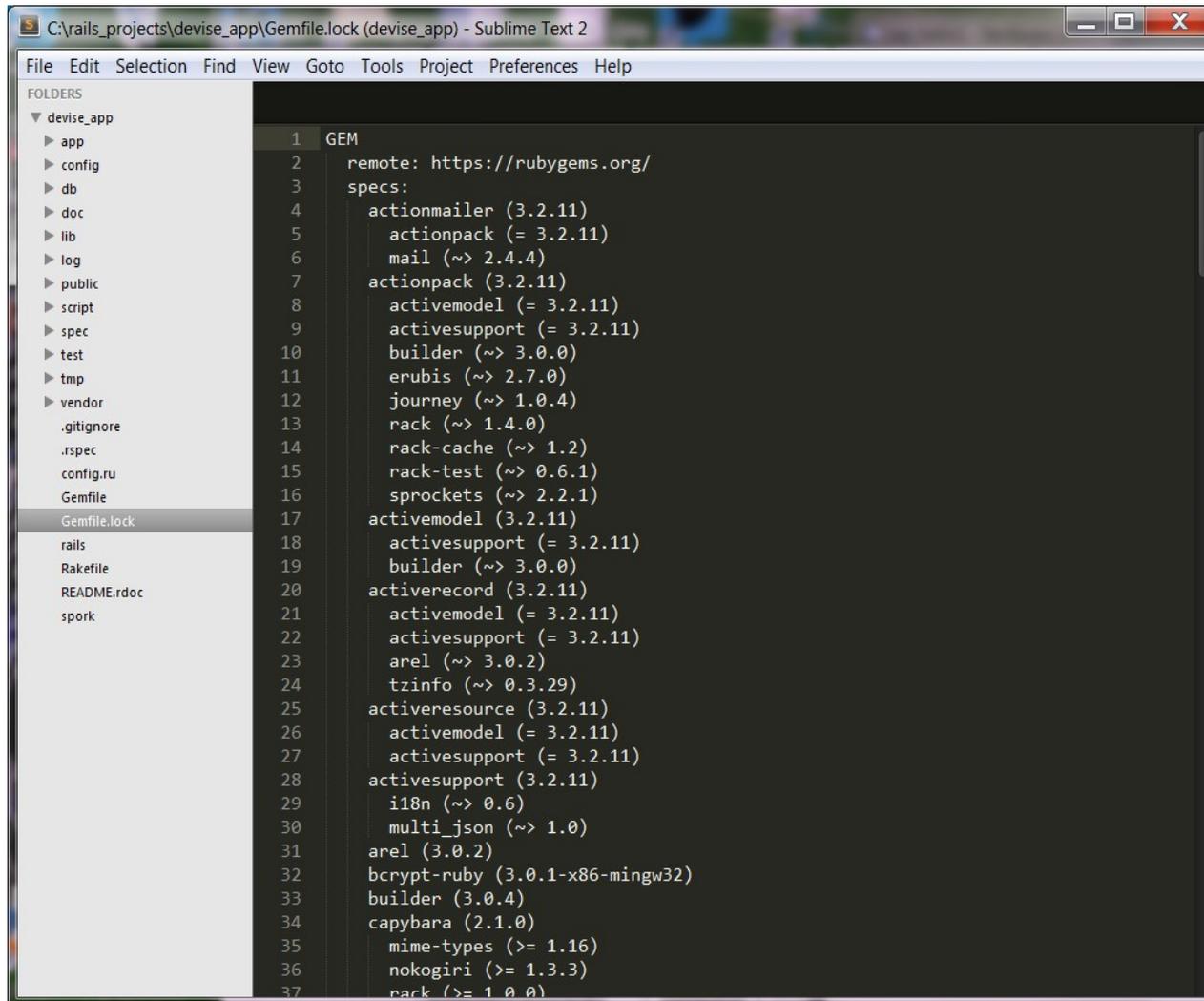
The screenshot shows a Sublime Text 2 editor window titled "C:\rails_projects\helloworld\Gemfile (helloworld) - Sublime Text 2". The interface includes a menu bar (File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, Help) and a left sidebar with a "FOLDERS" panel. The sidebar lists the project structure: helloworld (expanded) with subfolders app, bin, config, db, lib, log, public, test, tmp, vendor, and files .gitignore, config.ru, Gemfile, Gemfile.lock, Rakefile, and README.rdoc. The main editor area displays the content of the Gemfile, with line numbers 1 through 32. The first line, `1 source 'https://rubygems.org'`, is circled in orange. The rest of the file contains various gem declarations and comments, such as `gem 'rails', '4.1.1'`, `gem 'sqlite3'`, `gem 'sass-rails', '~> 4.0.3'`, `gem 'uglifier', '>= 1.3.0'`, `gem 'coffee-rails', '~> 4.0.0'`, `gem 'jquery-rails'`, `gem 'turbolinks'`, `gem 'jbuilder', '~> 2.0'`, `gem 'sdoc', '~> 0.4.0', group: :doc`, and `gem 'unicorn'`.

```
1 source 'https://rubygems.org'
2
3
4 # Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
5 gem 'rails', '4.1.1'
6 # Use sqlite3 as the database for Active Record
7 gem 'sqlite3'
8 # Use SCSS for stylesheets
9 gem 'sass-rails', '~> 4.0.3'
10 # Use Uglifier as compressor for JavaScript assets
11 gem 'uglifier', '>= 1.3.0'
12 # Use CoffeeScript for .js.coffee assets and views
13 gem 'coffee-rails', '~> 4.0.0'
14 # See https://github.com/sstephenson/execjs#readme for more
15 # gem 'therubyracer', platforms: :ruby
16
17 # Use jquery as the JavaScript library
18 gem 'jquery-rails'
19 # Turbolinks makes following links in your web application
20 gem 'turbolinks'
21 # Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder
22 gem 'jbuilder', '~> 2.0'
23 # bundle exec rake doc:rails generates the API under doc/api
24 gem 'sdoc', '~> 0.4.0', group: :doc
25
26 # Use ActiveRecord has_secure_password
27 # gem 'bcrypt', '~> 3.1.7'
28
29 # Use unicorn as the app server
30 gem 'unicorn'
31
32 # Use Capistrano for deployment
```

What is Gemfile.lock?

- When you run “bundle install”, Bundler will create *Gemfile.lock* if it does not exist already
 - > *Gemfile.lock* specifies all gems used in the application along with their versions
- Bundler uses this file in all subsequent 'bundle install', which guarantees that you always use the same exact version of gems, even as your application moves across machines
- You SHOULD check your *Gemfile.lock* into version control
 - > If you do not, meaning if there is no Gemfile.lock, every machine that checks out your repository (including your production server) will resolve all dependencies again, which could result in different versions of third-party code being used if any of the gems in the *Gemfile* or any of their dependencies have been updated

Gemfile.lock Example



```
1 GEM
2 remote: https://rubygems.org/
3 specs:
4   actionmailer (3.2.11)
5     actionpack (= 3.2.11)
6     mail (~> 2.4.4)
7   actionpack (3.2.11)
8     activemodel (= 3.2.11)
9     activesupport (= 3.2.11)
10    builder (~> 3.0.0)
11    erubis (~> 2.7.0)
12    journey (~> 1.0.4)
13    rack (~> 1.4.0)
14    rack-cache (~> 1.2)
15    rack-test (~> 0.6.1)
16    sprockets (~> 2.2.1)
17  activemodel (3.2.11)
18    activesupport (= 3.2.11)
19    builder (~> 3.0.0)
20  activerecord (3.2.11)
21    activemodel (= 3.2.11)
22    activesupport (= 3.2.11)
23    arel (~> 3.0.2)
24    tzinfo (~> 0.3.29)
25  activeresource (3.2.11)
26    activemodel (= 3.2.11)
27    activesupport (= 3.2.11)
28  activesupport (3.2.11)
29    i18n (~> 0.6)
30    multi_json (~> 1.0)
31  arel (3.0.2)
32  bcrypt-ruby (3.0.1-x86-mingw32)
33  builder (3.0.4)
34  capybara (2.1.0)
35    mime-types (>= 1.16)
36    nokogiri (>= 1.3.3)
37    rack (>= 1.0.0)
```

Bundler

What is a Bundler?

- Handles dependency management in Rails 3 and Rails 4 applications
 - > Performs dependency resolution on the complete list of gems specified in the Gemfile “all at once”
 - > Solves the “dependency conflict found too late” problem of Rails 2 (In Rails 2, dependency is resolved “one at a time”)

Bundler commands (used with app)

- “bundle install” (or just “bundle”)
 - > Install newly added gems specified in the “Gemfile”
- What it does
 - > If this is the first time you run “bundle install” (and a “Gemfile.lock” does not exist), bundler will fetch all gems, resolve dependencies, and install them locally, then creates “Gemfile.lock”
 - > If “Gemfile.lock” does exist, and you have not updated your “Gemfile”, bundler use the gems specified in the “Gemfile.lock”
 - > If “Gemfile.lock” does exist, and you have updated your “Gemfile”, bundler will use the dependencies in the “Gemfile.lock” for all gems that you did not update, but will re-resolve the dependencies of gems that you did update.

Bundler commands (used with app)

- “bundle show [gemname]”
 - > Displays where a bundled gem is installed
- “bundle update”
 - > Install newly added gems in the Gemfile (same as “bundle install”)
 - > If existing gems don't have version, it will upgrade to whatever latest (difference from “bundle install”)
 - > If existing gems have version controlled with ~>, it will upgrade to the latest at the final digit, the patch version (difference from “bundle install”)
 - > 'my_gem', '~> 2.1.0' - “bundle update” will check if newer version of 2.1.x is available

Bundler commands (used with app)

- “bundle package”
 - > Package up all gems in the “vendor/cache” directory inside app
 - > “bundle install” will use gems in the package instead of “rubygems.org”
 - > Used to avoid external dependencies at deploy time
- “bundle outdated”
 - > Show all of the outdated gems in the current bundle

“bundle exec <rails command>”

- “bundle exec <rails command>” executes a command in the context of your bundle
 - > Uses the gem versions specified in your “Gemfile” file (instead of the latest gem version installed in the system)
 - > Most of the time, running “bundle exec <command>” has the same results as if you just ran “<command>”, especially if you have the same gems installed system wide as in your Gemfile file
 - > Using “bundle exec <command>” guarantees that the program is run with the environment specified in the Gemfile, which hopefully means it is the environment that the creators of the program want it to be run in, which hopefully means it should run correctly no matter what weird setup you have on your computer
- Example
 - > bundle exec rake --tasks

Lab:

Exercise 2: Bundler
5524_rails4_configuration.zip

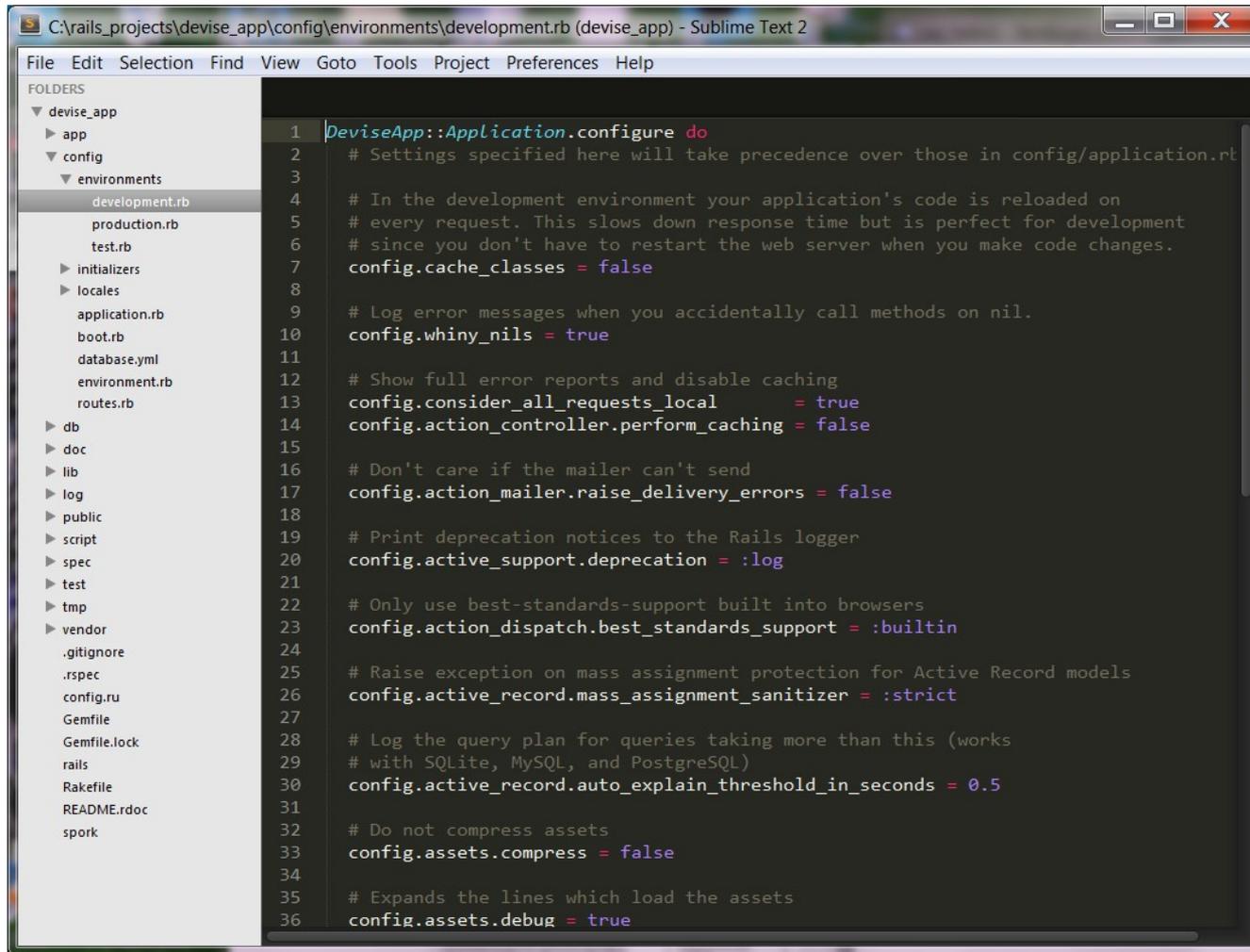


Environments

Environments

- By default Rails ships with three environments:
 - > "development"
 - > "test"
 - > "production"
- You can create your own custom environment, for example, "staging"
 - > Create a file called *config/environments/staging.rb*
- Start rails app with your custom environment
 - > *rails server -e staging*
- Start Rails console with your custom environment
 - > *rails console staging*

Development Environment



```
C:\rails_projects\devise_app\config\environments\development.rb (devise_app) - Sublime Text 2
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▼ devise_app
    ▶ app
    ▼ config
      ▼ environments
        development.rb
        production.rb
        test.rb
      ▶ initializers
      ▶ locales
      application.rb
      boot.rb
      database.yml
      environment.rb
      routes.rb
    ▶ db
    ▶ doc
    ▶ lib
    ▶ log
    ▶ public
    ▶ script
    ▶ spec
    ▶ test
    ▶ tmp
    ▶ vendor
    .gitignore
    .rspec
    config.ru
    Gemfile
    Gemfile.lock
    rails
    Rakefile
    README.rdoc
    spork

1 DeviseApp::Application.configure do
2   # Settings specified here will take precedence over those in config/application.rb
3
4   # In the development environment your application's code is reloaded on
5   # every request. This slows down response time but is perfect for development
6   # since you don't have to restart the web server when you make code changes.
7   config.cache_classes = false
8
9   # Log error messages when you accidentally call methods on nil.
10  config.whiny_nils = true
11
12  # Show full error reports and disable caching
13  config.consider_all_requests_local       = true
14  config.action_controller.perform_caching = false
15
16  # Don't care if the mailer can't send
17  config.action_mailer.raise_delivery_errors = false
18
19  # Print deprecation notices to the Rails logger
20  config.active_support.deprecation = :log
21
22  # Only use best-standards-support built into browsers
23  config.action_dispatch.best_standards_support = :builtin
24
25  # Raise exception on mass assignment protection for Active Record models
26  config.active_record.mass_assignment_sanitizer = :strict
27
28  # Log the query plan for queries taking more than this (works
29  # with SQLite, MySQL, and PostgreSQL)
30  config.active_record.auto_explain_threshold_in_seconds = 0.5
31
32  # Do not compress assets
33  config.assets.compress = false
34
35  # Expands the lines which load the assets
36  config.assets.debug = true
```

Lab:

Exercise 3: Environments 5524_rails4_configuration.zip

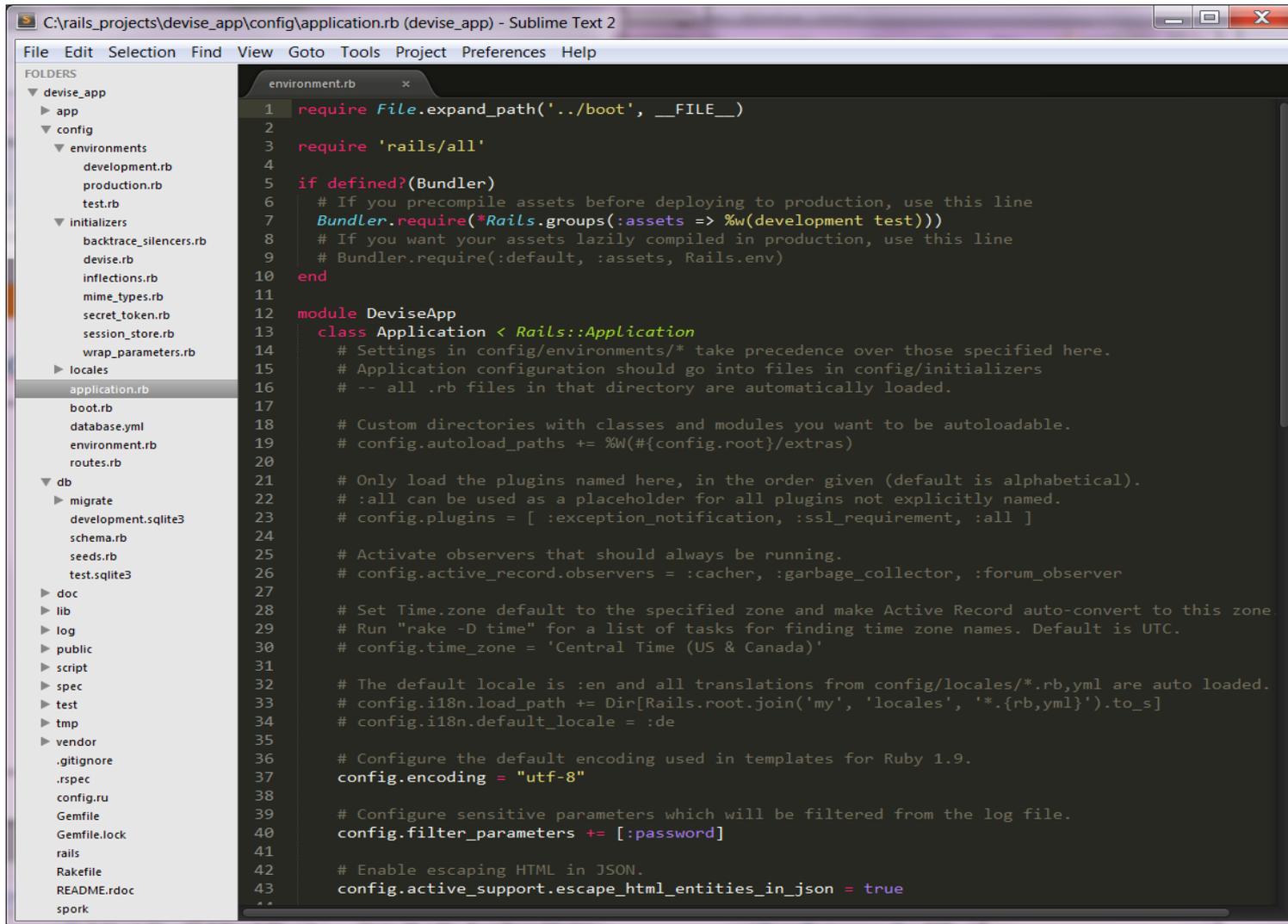


Configuration

Location for Initialization Code

- Rails offers four standard spots to place initialization code:
 - > `config/application.rb`
 - > Environment-specific configuration files
 - > Initializers
 - > After-initializers

config/application.rb

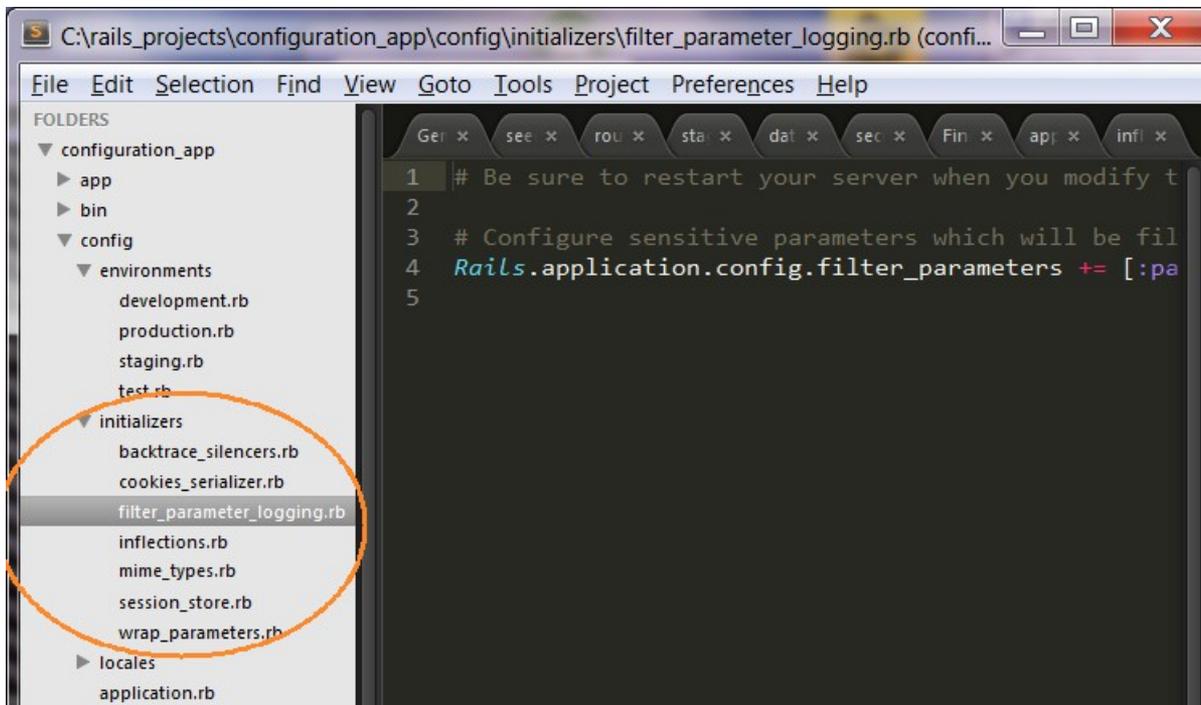


```
1 require File.expand_path('../boot', __FILE__)
2
3 require 'rails/all'
4
5 if defined?(Bundler)
6   # If you precompile assets before deploying to production, use this line
7   Bundler.require(*Rails.groups(:assets => %w(development test)))
8   # If you want your assets lazily compiled in production, use this line
9   # Bundler.require(:default, :assets, Rails.env)
10 end
11
12 module DeviseApp
13   class Application < Rails::Application
14     # Settings in config/environments/* take precedence over those specified here.
15     # Application configuration should go into files in config/initializers
16     # -- all .rb files in that directory are automatically loaded.
17
18     # Custom directories with classes and modules you want to be autoloadable.
19     # config.autoload_paths += %W(#{config.root}/extras)
20
21     # Only load the plugins named here, in the order given (default is alphabetical).
22     # :all can be used as a placeholder for all plugins not explicitly named.
23     # config.plugins = [ :exception_notification, :ssl_requirement, :all ]
24
25     # Activate observers that should always be running.
26     # config.active_record.observers = :cache, :garbage_collector, :forum_observer
27
28     # Set Time.zone default to the specified zone and make Active Record auto-convert to this zone.
29     # Run "rake -D time" for a list of tasks for finding time zone names. Default is UTC.
30     # config.time_zone = 'Central Time (US & Canada)'
31
32     # The default locale is :en and all translations from config/locales/*.rb,yml are auto loaded.
33     # config.i18n.load_path += Dir[Rails.root.join('my', 'locales', '*.{rb,yml}').to_s]
34     # config.i18n.default_locale = :de
35
36     # Configure the default encoding used in templates for Ruby 1.9.
37     config.encoding = "utf-8"
38
39     # Configure sensitive parameters which will be filtered from the log file.
40     config.filter_parameters += [:password]
41
42     # Enable escaping HTML in JSON.
43     config.active_support.escape_html_entities_in_json = true
44   end
45 end
```

Initializers

Initializer Files

- After loading the framework and any gems in your application, Rails turns to loading initializers
- An initializer is any Ruby file stored under *config/initializers* in your application – you can add your own initializers here



Lab:

Exercise 4: Initializers
5524_rails4_configuration.zip

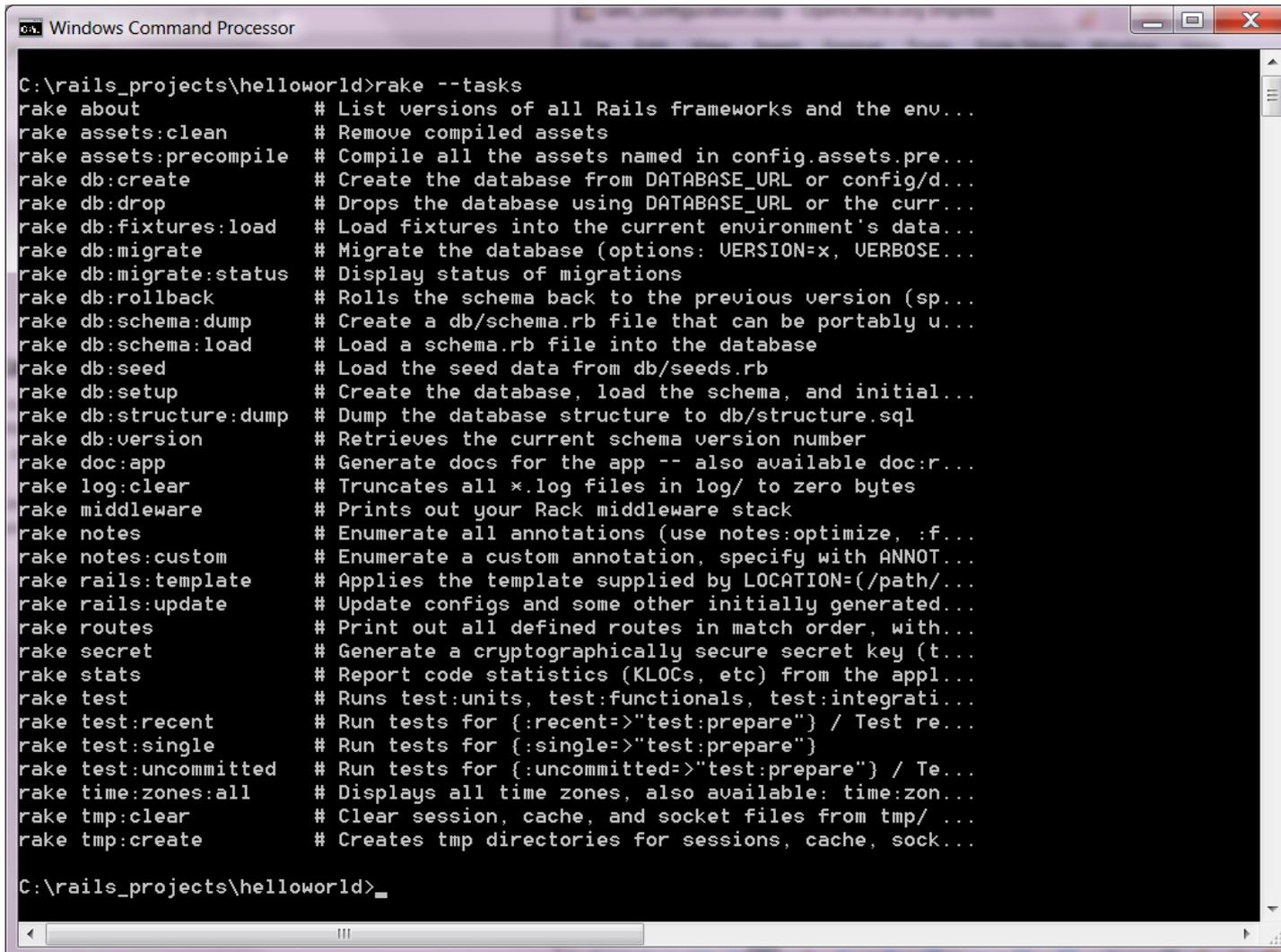


Rake

What is “Rake”?

- Rake is Ruby Make, a standalone Ruby utility that replaces the Unix utility 'make', and uses a '*Rakefile*' and *.rake* files to build up a list of tasks
- In Rails, Rake is used for common administration tasks, especially sophisticated ones that build off of each other

Ready to use “Rake”tasks



```
C:\rails_projects\helloworld>rake --tasks
rake about                # List versions of all Rails frameworks and the env...
rake assets:clean         # Remove compiled assets
rake assets:precompile    # Compile all the assets named in config.assets.pre...
rake db:create            # Create the database from DATABASE_URL or config/d...
rake db:drop              # Drops the database using DATABASE_URL or the curr...
rake db:fixtures:load     # Load fixtures into the current environment's data...
rake db:migrate           # Migrate the database (options: VERSION=x, VERBOSE...
rake db:migrate:status    # Display status of migrations
rake db:rollback          # Rolls the schema back to the previous version (sp...
rake db:schema:dump       # Create a db/schema.rb file that can be portably u...
rake db:schema:load       # Load a schema.rb file into the database
rake db:seed              # Load the seed data from db/seeds.rb
rake db:setup             # Create the database, load the schema, and initial...
rake db:structure:dump    # Dump the database structure to db/structure.sql
rake db:version           # Retrieves the current schema version number
rake doc:app              # Generate docs for the app -- also available doc:r...
rake log:clear            # Truncates all *.log files in log/ to zero bytes
rake middleware           # Prints out your Rack middleware stack
rake notes                # Enumerate all annotations (use notes:optimize, :f...
rake notes:custom         # Enumerate a custom annotation, specify with ANNOT...
rake rails:template       # Applies the template supplied by LOCATION=(/path/...
rake rails:update         # Update configs and some other initially generated...
rake routes               # Print out all defined routes in match order, with...
rake secret               # Generate a cryptographically secure secret key (t...
rake stats                # Report code statistics (KLOCs, etc) from the appl...
rake test                 # Runs test:units, test:functionals, test:integrati...
rake test:recent          # Run tests for {:recent=>"test:prepare"} / Test re...
rake test:single          # Run tests for {:single=>"test:prepare"}
rake test:uncommitted    # Run tests for {:uncommitted=>"test:prepare"} / Te...
rake time:zones:all       # Displays all time zones, also available: time:zon...
rake tmp:clear            # Clear session, cache, and socket files from tmp/ ...
rake tmp:create           # Creates tmp directories for sessions, cache, sock...

C:\rails_projects\helloworld>_
```

Creating Custom Rake Tasks

- Custom rake tasks have a .rake extension and are placed in ./lib/tasks directory

```
namespace :greeting do
  desc "Some greeting"
  task :say_hello do
    puts "Hello, JPassion.com!"
  end
  task :say_goodbye do
    puts "Goodbye, JPassion.com!!!"
  end
end
```

- Then you can invoke the task

```
rake greeting:say_hello
rake greeting:say_goodbye
```

Lab:

Exercise 5: Rake 5553_rails4_configuration.zip



Code with Passion!
JPassion.com

