A.9 Checklist: Starting a New Rails App

Throughout the book we recommend several tools for developing, testing, deploying, and monitoring the code quality of your app. In this section, we pull together in one place a step-by-step list for creating a new app that takes advantage of all these tools. This section will only make sense after you have read all the referenced sections, so use it as a reference and don't worry if you don't understand all the steps now. Steps are annotated with the section number(s) in which the tool or concept is first introduced.

Set up your app: (§4.1)

1. `rails -v` to ensure you're running the desired version of Rails. If not run `gem install rails -v` *x.x.x* with *x.x.x* set to the version you want; 3.2.19 for example.
2. `rails new` *appname* `-T` to create the new app. `-T` skips creating the `test` subdirectory used by the `Test::Unit` testing framework, since we recommend using RSpec instead.
3. `cd` *appname* to navigate into your new app's root directory. From now on, all shell commands should be issued from this directory.
4. Edit the `Gemfile` to lock the versions of Ruby and Rails, for example:

   http://pastebin.com/6NxFRNrM

   ```
   1 # in Gemfile:
   2 ruby '1.9.3'     # Ruby version you're running
   3 rails '3.2.19'   # Rails version for this app
   ```

   If you ended up changing the version(s) already present in the Gemfile, run `bundle install --without production` to make sure you have compatible versions of Rails and other gems.

5. Make sure your app runs by executing `rails server` and visiting `http://localhost:3000`. You should see the Rails welcome page.
6. `git init` to set up your app's root directory as a GitHub repo. (§A.6, Screencast A.6.1)

Connect your app to GitHub, CodeClimate, and Heroku:

1. Create a GitHub repo via GitHub's web interface, and do the initial commit and push of your new app's repo. (§A.7)
2. Point CodeClimate at your app's GitHub repo. (§9.5)
3. Make the changes necessary to deploy to production on Heroku. (§A.8)
4. Run `bundle install --without production` if you've changed your `Gemfile`. Commit the changes to `Gemfile` and `Gemfile.lock`. On future changes to the Gemfile, you can just say `bundle` with no arguments, since Bundler will remember the option to skip production gems. (§4.1)
5. `heroku apps:create` *appname* to create your new app on Heroku (§A.8)
6. `git push heroku master` to ensure the app deploys correctly. You should then be able to visit your app's Rails splash page at `http://`*appname*`.herokuapp.com`. At this point you can safely remove the default splash page: `git rm public/index.html`. (§A.8)

Set up your testing environment:

1. Add support in your Gemfile for Cucumber (§7.6), RSpec (§8.2), interactive debugging (§4.1), SimpleCov (§8.7), Autotest (§8.2), FactoryGirl (§8.5), Jasmine if you plan to use JavaScript (§6.7), and Metric-Fu to keep track of your code metrics:

   http://pastebin.com/y4MaVP72

```
1 # debugger is useful in development mode too
2 group :development, :test do
```

```
 3   gem 'debugger'
 4   gem 'jasmine-rails' # if you plan to use JavaScript/C
offeeScript
 5 end
 6 # setup Cucumber, RSpec, autotest support
 7 group :test do
 8   gem 'rspec-rails', '2.14'
 9   gem 'simplecov', :require => false
10   gem 'cucumber-rails', :require => false
11   gem 'cucumber-rails-training-wheels' # basic imperati
ve step defs
12   gem 'database_cleaner' # required by Cucumber
13   gem 'autotest-rails'
14   gem 'factory_girl_rails' # if using FactoryGirl
15   gem 'metric_fu'          # collect code metrics
16 end
```

(See Section 6.7 for additional gems to support fixtures and AJAX stubbing in your JavaScript tests.)

2. Run `bundle`, since you've changed your `Gemfile`. Commit the changes to `Gemfile` and `Gemfile.lock`.
3. If all is well, create the subdirectories and files used by RSpec, Cucumber, Jasmine, and if you're using them, the basic Cucumber imperative steps:

   http://pastebin.com/BvJvHezi

```
1 rails generate rspec:install
2 rails generate cucumber:install
3 rails generate cucumber_rails_training_wheels:install
4 rails generate jasmine_rails:install
```

4. If you're using SimpleCov, which we recommend, place the following lines at the *top* of `spec/spec_helper.rb` to enable it:

   http://pastebin.com/G5BV1efA

   ```
   1 # at TOP of spec/spec_helper.rb:
   2 require 'simplecov'
   3 SimpleCov.start
   ```

5. If you're using FactoryGirl to manage factories (§8.5), add its setup code:

   http://pastebin.com/VDnhECsQ

   ```
   1 # For RSpec, create this file as spec/support/factory_g
   irl.rb
   2 RSpec.configure do |config|
   3   config.include FactoryGirl::Syntax::Methods
   4 end
   ```

   http://pastebin.com/Wx7veG8E

   ```
   1 # For Cucumber, add at the end of features/support/env.
   rb:
   2 World(FactoryGirl::Syntax::Methods)
   ```

6. `git add` and then commit any files created or modified by these steps.
7. Ensure Heroku deployment still works: `git push heroku master`

You're now ready to create and apply the first migration (§4.2), then re-deploy to Heroku and apply the migration in production (`heroku run rake db:migrate`).

Add other useful Gems:

Some that we recommend include:

- `railroady` draws diagrams of your class relationships such as has-many, belongs-to, and so on (§5.3)
- `omniauth` adds portable third-party authentication (§5.2)
- `devise` adds user self-signup pages, and optionally works with `omniauth`