[1.2.2 Ruby, RubyGems, Rails, and Git](http://ruby.railstutorial.org/chapters/beginning" \l "sec-rubygems)

*Practically all the software in the world is either broken or very difficult to use. So users dread software. They’ve been trained that whenever they try to install something, or even fill out a form online, it’s not going to work. I dread installing stuff, and I have a Ph.D. in computer science.   
—Paul Graham, in Founders at Work by Jessica Livingston*

Now it’s time to install Ruby and Rails. I’ve done my best to cover as many bases as possible, but systems vary, and many things can go wrong during these steps. Be sure to Google the error message or consult the [Rails Tutorial help page](http://railstutorial.org/help) if you run into trouble. Also, there’s a new resource called [Install Rails](http://www.installrails.com/) from [One Month Rails](https://onemonthrails.com/) that might help you if you get stuck.

**Unless otherwise noted, you should use the exact versions of all software used in the tutorial, including Rails itself, if you want the same results.** Sometimes minor version differences will yield identical results, but you shouldn’t count on this, especially with respect to Rails versions. The main exception is Ruby itself: 1.9.3 and 2.0.0 are virtually identical for the purposes of this tutorial, so feel free to use either one.

[Rails Installer (Windows)](http://ruby.railstutorial.org/chapters/beginning#sec-rails_installer_windows)

Installing Rails on Windows used to be a real pain, but thanks to the efforts of the good people at[Engine Yard](http://engineyard.com/)—especially Dr. Nic Williams and Wayne E. Seguin—installing Rails and related software on Windows is now easy. If you are using Windows, go to [Rails Installer](http://railsinstaller.org/) and download the Rails Installer executable and view the excellent installation video. Double-click the executable and follow the instructions to install Git (so you can skip [Section 1.2.2.2](http://ruby.railstutorial.org/chapters/beginning#sec-install_git)), Ruby (skip [Section 1.2.2.3](http://ruby.railstutorial.org/chapters/beginning#sec-install_ruby)), RubyGems (skip [Section 1.2.2.4](http://ruby.railstutorial.org/chapters/beginning#sec-install_rubygems)), and Rails itself (skip [Section 1.2.2.5](http://ruby.railstutorial.org/chapters/beginning#sec-install_rails)). Once the installation has finished, you can skip right to the creation of the first application in [Section 1.2.3](http://ruby.railstutorial.org/chapters/beginning#sec-the_first_application).

Bear in mind that the Rails Installer might use a slightly different version of Rails from the one installed in [Section 1.2.2.5](http://ruby.railstutorial.org/chapters/beginning#sec-install_rails), which might cause incompatibilities. To fix this, I am currently working with Nic and Wayne to create a list of Rails Installers ordered by Rails version number.

[Install Git](http://ruby.railstutorial.org/chapters/beginning#sec-install_git)

Much of the Rails ecosystem depends in one way or another on a [version control system](http://en.wikipedia.org/wiki/Revision_control) called [Git](http://git-scm.com/)(covered in more detail in [Section 1.3](http://ruby.railstutorial.org/chapters/beginning#sec-version_control)). Because its use is ubiquitous, you should install Git even at this early stage; I suggest following the installation instructions for your platform at the [Installing Git section of *Pro Git*](http://www.git-scm.com/book/en/Getting-Started-Installing-Git).

[Install Ruby](http://ruby.railstutorial.org/chapters/beginning#sec-install_ruby)

The next step is to install Ruby. (This can be painful and error-prone, and I actually dread having to install new versions of Ruby, but unfortunately it’s the cost of doing business.)

It’s possible that your system already has Ruby installed. Try running

**$** ruby -v

to see the version number. Rails 4 requires Ruby 1.9 or later and on most systems works best with Ruby 2.0. (In particular, it won’t work Ruby 1.8.7.) This tutorial assumes that most readers are using Ruby 1.9.3 or 2.0.0, but Ruby 1.9.2 should work as well. **Note:** I’ve had reports from Windows users that Ruby 2.0 is sketchy, so I recommend using Ruby 1.9.3 if you’re on Windows.

As part of installing Ruby, if you are using OS X or Linux I strongly recommend using [Ruby Version Manager (RVM)](http://rvm.io/) or [rbenv](https://github.com/sstephenson/rbenv), which allow you to install and manage multiple versions of Ruby on the same machine. (The [Pik](http://github.com/vertiginous/pik) project accomplishes a similar feat on Windows.) This is particularly important if you want to run different versions of Ruby or Rails on the same machine. Unfortunately, RVM and rbenv can’t be used on the same system simultaneously, and since I’ve been using RVM longer that’s the one I use in this tutorial. I hear great things about rbenv, though, so you should feel free to use that if you already know it or if you have access to a local rbenv expert.

As a prerequisite, OS X users may need to install the Xcode developer tools. To avoid the (huge) full installation, I recommend the much smaller [Command Line Tools for Xcode](https://developer.apple.com/downloads/).[12](http://ruby.railstutorial.org/chapters/beginning#fn-1_12)

To get started with the Ruby installation, first [install RVM](http://rvm.io/rvm/install/):

**$** curl -L https://get.rvm.io | bash -s

(If you have RVM installed, you should run

**$** rvm get stable

to ensure that you have the latest version.)

You can then get Ruby set up by examining the requirements for installing it:

**$** rvm requirements

On my system, I had to install the following (using [Homebrew](http://mxcl.github.com/homebrew/), a package management system for OS X):

**$** brew install libtool libxslt libksba openssl

On Linux, you can accomplish similar things with **apt-get** or **yum**.

I also had to install a [YAML](https://en.wikipedia.org/wiki/YAML) library:

**#** For Mac with Homebrew

**$** brew install libyaml

**#** For Debian-based Linux systems

**$** apt-get install libyaml-dev

**#** For Fedora/CentOS/RHEL Linux systems

**$** yum install libyaml-devel

Finally, I needed to tell RVM where OpenSSL was located when installing Ruby 2.0.0:

**$** rvm install 2.0.0 --with-openssl-dir=$HOME/.rvm/usr

<wait a while>

On some systems, especially on Macs using Homebrew, the location of OpenSSL may be different, and you might have to run this command instead:

**$** rvm install 2.0.0 --with-openssl-dir=$HOME/.rvm/opt/openssl

<wait a while>

Unfortunately, lots of things can go wrong along the way. I’ve done my best to cover some of the most common cases, but the only general solution is web searches and determination.

After installing Ruby, you should configure your system for the other software needed to run Rails applications. This typically involves installing *gems*, which are self-contained packages of Ruby code. Since gems with different version numbers sometimes conflict, it is often convenient to create separate *gemsets*, which are self-contained bundles of gems. For the purposes of this tutorial, I suggest creating a gemset called **railstutorial\_rails\_4\_0**:

**$** rvm use 2.0.0@railstutorial\_rails\_4\_0 --create --default

Using /Users/mhartl/.rvm/gems/ruby-2.0.0-p0 with gemset railstutorial\_rails\_4\_0

This command creates (--create) the gemset **railstutorial\_rails\_4\_0** associated with Ruby 2.0.0 while arranging to start using it immediately (use) and setting it as the default (--default) gemset, so that any time we open a new terminal window the**2.0.0@railstutorial\_rails\_4\_0** Ruby/gemset combination is automatically selected. RVM supports a large variety of commands for manipulating gemsets; see the documentation at[http://rvm.beginrescueend.com/gemsets/](http://rvm.io/gemsets/). If you ever get stuck with RVM, running commands like these should help you get your bearings:

**$** rvm help

**$** rvm gemset help

For more information on RVM, I also recommend taking a look at the article [Ruby Version Manager (RVM) Overview for Rails Newbs](http://strandcode.com/2013/07/11/ruby-version-manager-rvm-overview-for-rails-newbs/).[13](http://ruby.railstutorial.org/chapters/beginning#fn-1_13)

[Install RubyGems](http://ruby.railstutorial.org/chapters/beginning#sec-install_rubygems)

RubyGems is a package manager for Ruby projects, and there are many useful libraries (including Rails) available as Ruby packages, or *gems*. Installing RubyGems should be easy once you install Ruby. In fact, if you have [installed RVM](http://rvm.io/rvm/install/), you already have RubyGems, since RVM includes it automatically:

**$** which gem

/Users/mhartl/.rvm/rubies/ruby-2.0.0-p0/bin/gem

If you don’t already have it, you should [download RubyGems](http://rubyforge.org/frs/?group_id=126), extract it, and then go to the**rubygems** directory and run the setup program:

**$** ruby setup.rb

(If you get a permissions error here, recall from [Section 1.1.3](http://ruby.railstutorial.org/chapters/beginning#sec-conventions) that you may have to use **sudo**.)

If you already have RubyGems installed, you should make sure your system uses the version used in this tutorial:

**$** gem update --system 2.1.9

Freezing your system to this particular version will help prevent conflicts as RubyGems changes in the future.

When installing gems, by default RubyGems generates two different kinds of documentation (called ri and rdoc), but many Ruby and Rails developers find that the time to build them isn’t worth the benefit. (Many programmers rely on online documentation instead of the native ri and rdoc documents.) To prevent the automatic generation of the documentation, I recommend making a gem configuration file called **.gemrc** in your home directory as in [Listing 1.1](http://ruby.railstutorial.org/chapters/beginning#code-create_gemrc) with the line in[Listing 1.2](http://ruby.railstutorial.org/chapters/beginning#code-gemrc). (The tilde “~” means “home directory”, while the dot . in **.gemrc** makes the file hidden, which is a common convention for configuration files. )

**Listing 1.1.** Creating a gem configuration file.

**$** subl ~/.gemrc

Here **subl** is the command-line command to launch Sublime Text on OS X, which you can set up using the [Sublime Text 2 documentation for the OS X command line](http://www.sublimetext.com/docs/2/osx_command_line.html). If you’re on a different platform, or if you’re using a different editor, you should replace this command as necessary (i.e., by double-clicking the application icon or by using an alternate command such as **mate**, **vim**, **gvim**, or**mvim**). For brevity, throughout the rest of this tutorial I’ll use **subl** as a shorthand for “open with your favorite text editor.”

**Listing 1.2.** Suppressing the ri and rdoc documentation in **.gemrc**.

install: --no-rdoc --no-ri

update: --no-rdoc --no-ri

[Install Rails](http://ruby.railstutorial.org/chapters/beginning#sec-install_rails)

Once you’ve installed RubyGems, installing Rails should be easy. This tutorial standardizes on Rails 4.0, which we can install as follows:

**$** gem install rails --version 4.0.2

To check your Rails installation, run the following command to print out the version number:

**$** rails -v

Rails 4.0.2

*Note:* If you installed Rails using the Rails Installer in [Section 1.2.2.1](http://ruby.railstutorial.org/chapters/beginning#sec-rails_installer_windows), there might be slight version differences. As of this writing, those differences are not relevant, but in the future, as the current Rails version diverges from the one used in this tutorial, these differences may become significant. I am currently working with Engine Yard to create links to specific versions of the Rails Installer.

If you’re running Linux, you might have to install a couple of other packages at this point:

**$** sudo apt-get install libxslt-dev libxml2-dev libsqlite3-dev *# Linux only*

[1.2.3 The first application](http://ruby.railstutorial.org/chapters/beginning#sec-the_first_application)

Virtually all Rails applications start the same way, by running **rails new** command. This handy command creates a skeleton Rails application in a directory of your choice. To get started, make a directory for your Rails projects and then run **rails new** to make the first application ([Listing 1.3](http://ruby.railstutorial.org/chapters/beginning#code-rails_command)):

**Listing 1.3.** Running **rails new** to generate a new application.

**$** mkdir rails\_projects

**$** cd rails\_projects

**$** rails new first\_app

create

create README.rdoc

create Rakefile

create config.ru

create .gitignore

create Gemfile

create app

create app/assets/javascripts/application.js

create app/assets/stylesheets/application.css

create app/controllers/application\_controller.rb

.

.

.

create test/test\_helper.rb

create tmp/cache

create tmp/cache/assets

create vendor/assets/javascripts

create vendor/assets/javascripts/.keep

create vendor/assets/stylesheets

create vendor/assets/stylesheets/.keep

run bundle install

.

.

.

Your bundle is complete! Use `bundle show [gemname]` to see where a bundled

gem is installed.

As seen at the end of [Listing 1.3](http://ruby.railstutorial.org/chapters/beginning#code-rails_command), running **rails new** automatically runs the **bundle install**command after the file creation is done. If that step doesn’t work right now,

don’t worry; follow the steps in [Section 1.2.4](http://ruby.railstutorial.org/chapters/beginning#sec-bundler) and you should be able to get it to work.

Notice how many files and directories the **rails** command creates. This standard directory and file structure ([Figure 1.2](http://ruby.railstutorial.org/chapters/beginning#fig-directory_structure_rails)) is one of the many advantages of Rails; it immediately gets you from zero to a functional (if minimal) application. Moreover, since the structure is common to all Rails apps, you can immediately get your bearings when looking at someone else’s code. A summary of the default Rails files appears in [Table 1.1](http://ruby.railstutorial.org/chapters/beginning#table-rails_directory_structure); we’ll learn about most of these files and directories throughout the rest of this book. In particular, starting in [Section 5.2.1](http://ruby.railstutorial.org/chapters/filling-in-the-layout#sec-the_asset_pipeline) we’ll discuss the **app/assets** directory, part of the *asset pipeline* that makes it easier than ever to organize and deploy assets such as cascading style sheets and JavaScript files.

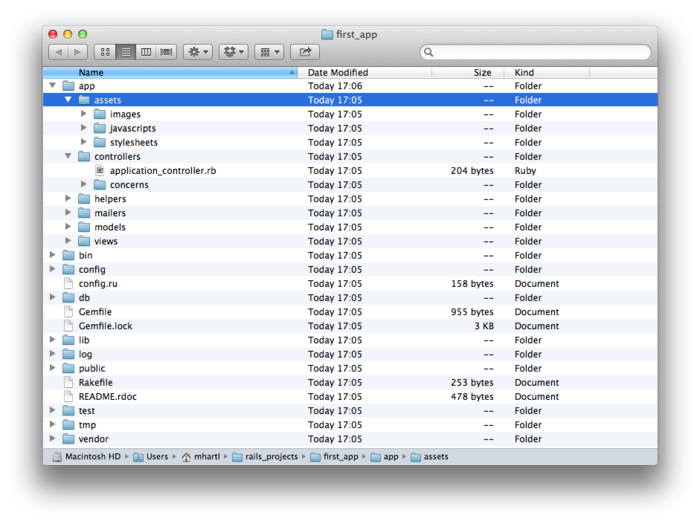


Figure 1.2: The directory structure for a newly created Rails app. [(full size)](http://railstutorial.org/images/figures/directory_structure_rails_4_0-full.png)

|  |  |
| --- | --- |
| **File/Directory** | **Purpose** |
| **app/** | Core application (app) code, including models, views, controllers, and helpers |
| **app/assets** | Applications assets such as cascading style sheets (CSS), JavaScript files, and images |
| **bin/** | Binary executable files |
| **config/** | Application configuration |
| **db/** | Database files |
| **doc/** | Documentation for the application |
| **lib/** | Library modules |
| **lib/assets** | Library assets such as cascading style sheets (CSS), JavaScript files, and images |
| **log/** | Application log files |
| **public/** | Data accessible to the public (e.g., web browsers), such as error pages |
| **bin/rails** | A program for generating code, opening console sessions, or starting a local server |
| **test/** | Application tests (made obsolete by the **spec/** directory in [Section 3.1](http://ruby.railstutorial.org/chapters/static-pages#sec-static_pages)) |
| **tmp/** | Temporary files |
| **vendor/** | Third-party code such as plugins and gems |
| **vendor/assets** | Third-party assets such as cascading style sheets (CSS), JavaScript files, and images |
| **README.rdoc** | A brief description of the application |
| **Rakefile** | Utility tasks available via the **rake** command |
| **Gemfile** | Gem requirements for this app |
| **Gemfile.lock** | A list of gems used to ensure that all copies of the app use the same gem versions |
| **config.ru** | A configuration file for [Rack middleware](http://rack.rubyforge.org/doc/) |
| **.gitignore** | Patterns for files that should be ignored by Git |

Table 1.1: A summary of the default Rails directory structure.

[1.2.4 Bundler](http://ruby.railstutorial.org/chapters/beginning#sec-bundler)

After creating a new Rails application, the next step is to use *Bundler* to install and include the gems needed by the app. As noted briefly in [Section 1.2.3](http://ruby.railstutorial.org/chapters/beginning#sec-the_first_application), Bundler is run automatically (via **bundle install**) by the **rails** command, but in this section we’ll make some changes to the default application gems and run Bundler again. This involves opening the **Gemfile** with your favorite text editor:

**$** cd first\_app/

**$** subl Gemfile

The result should look something like [Listing 1.4](http://ruby.railstutorial.org/chapters/beginning#code-default_gemfile). The code in this file is Ruby, but don’t worry at this point about the syntax; [Chapter 4](http://ruby.railstutorial.org/chapters/rails-flavored-ruby#top) will cover Ruby in more depth.

**Listing 1.4.** The default **Gemfile** in the **first\_app** directory.

source 'https://rubygems.org'

*# Use sqlite3 as the database for Active Record*

gem 'sqlite3'

*# Use SCSS for stylesheets*

gem 'sass-rails', '~> 4.0.1'

*# Use Uglifier as compressor for JavaScript assets*

gem 'uglifier', '>= 1.3.0'

*# Use CoffeeScript for .js.coffee assets and views*

gem 'coffee-rails', '~> 4.0.0'

*# See https://github.com/sstephenson/execjs#readme for more supported runtimes*

*# gem 'therubyracer', platforms: :ruby*

*# Use jquery as the JavaScript library*

gem 'jquery-rails'

*# Turbolinks makes following links in your web application faster.*

*# Read more: https://github.com/rails/turbolinks*

gem 'turbolinks'

*# Build JSON APIs with ease. Read more: https://github.com/rails/jbuilder*

gem 'jbuilder', '~> 1.0.1'

group :doc **do**

*# bundle exec rake doc:rails generates the API under doc/api.*

gem 'sdoc', require: false

**end**

*# Use ActiveModel has\_secure\_password*

*# gem 'bcrypt-ruby', '~> 3.1.2'*

*# Use unicorn as the app server*

*# gem 'unicorn'*

*# Use Capistrano for deployment*

*# gem 'capistrano', group: :development*

*# Use debugger*

*# gem 'debugger', group: [:development, :test]*

Many of these lines are commented out with the hash symbol **#**; they are there to show you some commonly needed gems and to give examples of the Bundler syntax. For now, we won’t need any gems other than the defaults.

Unless you specify a version number to the **gem** command, Bundler will automatically install the latest version of the gem. Unfortunately, gem updates often cause minor but potentially confusing breakage, so in this tutorial we’ll include explicit version numbers known to work, as seen in[Listing 1.5](http://ruby.railstutorial.org/chapters/beginning#code-gemfile_sqlite_version) (which also omits the commented-out lines from [Listing 1.4](http://ruby.railstutorial.org/chapters/beginning#code-default_gemfile)).

**Listing 1.5.** A **Gemfile** with an explicit version for each Ruby gem.

source 'https://rubygems.org'

ruby '2.0.0'

*#ruby-gemset=railstutorial\_rails\_4\_0*

gem 'rails', '4.0.2'

group :development **do**

gem 'sqlite3', '1.3.8'

**end**

gem 'sass-rails', '4.0.1'

gem 'uglifier', '2.1.1'

gem 'coffee-rails', '4.0.1'

gem 'jquery-rails', '3.0.4'

gem 'turbolinks', '1.1.1'

gem 'jbuilder', '1.0.2'

group :doc **do**

gem 'sdoc', '0.3.20', require: false

**end**

[Listing 1.5](http://ruby.railstutorial.org/chapters/beginning#code-gemfile_sqlite_version) adds the lines

ruby '2.0.0'

*#ruby-gemset=railstutorial\_rails\_4\_0*

identifying the version of Ruby expected by the application (especially useful when deploying applications ([Section 1.4](http://ruby.railstutorial.org/chapters/beginning#sec-deploying))), along with the RVM gemset ([Section 1.2.2.3](http://ruby.railstutorial.org/chapters/beginning#sec-install_ruby)). Because the gemset line starts with **#**, which is the Ruby comment character, it will be ignored if you aren’t using RVM, but if you are RVM will conveniently use the right Ruby version/gemset combination upon entering the application directory. (If you are using a version of Ruby other than 2.0.0, you should change the Ruby version line accordingly.)

The updated **Gemfile** also changes the line for jQuery, the default JavaScript library used by Rails, from

gem 'jquery-rails'

to

gem 'jquery-rails', '3.0.4'

We’ve also changed

gem 'sqlite3'

to

group :development **do**

gem 'sqlite3', '1.3.8'

**end**

which forces Bundler to install version **1.3.8** of the **sqlite3** gem. Note that we’ve also taken this opportunity to arrange for the gem to be included only in a development environment ([Section 7.1.1](http://ruby.railstutorial.org/chapters/sign-up#sec-rails_environments)), which prevents potential conflicts with the database used by Heroku ([Section 1.4](http://ruby.railstutorial.org/chapters/beginning#sec-deploying)).

[Listing 1.5](http://ruby.railstutorial.org/chapters/beginning#code-gemfile_sqlite_version) also changes a few other lines, converting

*# Use SCSS for stylesheets*

gem 'sass-rails', '~> 4.0.0'

*# Use Uglifier as compressor for JavaScript assets*

gem 'uglifier', '>= 1.3.0'

*# Use CoffeeScript for .js.coffee assets and views*

gem 'coffee-rails', '~> 4.0.0'

to

gem 'sass-rails', '4.0.1'

gem 'uglifier', '2.1.1'

gem 'coffee-rails', '4.0.1'

The syntax

gem 'uglifier', '>= 1.3.0'

installs the latest version of the **uglifier** gem (which handles file compression for the asset pipeline) as long as it’s greater than or equal to version **1.3.0**—even if it’s, say, version **7.2**. Meanwhile, the code

gem 'coffee-rails', '~> 4.0.0'

installs the gem **coffee-rails** (also needed by the asset pipeline) as long as it’s newer than version **4.0.0** but not newer than **4.1**. In other words, the >= notation always installs the latest gem when you run **bundle install**, whereas the ~> 4.0.0 notation only installs updated gems representing minor point releases (e.g., from **4.0.0** to **4.0.1**), but not major point releases (e.g., from **4.0** to **4.1**). Unfortunately, experience shows that even minor point releases can break things, so for the *Rails Tutorial* we’ll err on the side of caution by including exact version numbers for virtually all gems. You are welcome to use the most up-to-date version of any gem, including using the ~> construction in the **Gemfile** (which I generally recommend for more advanced users), but be warned that this may cause the tutorial to act unpredictably.

Once you’ve assembled the proper **Gemfile**, install the gems using **bundle update**[14](http://ruby.railstutorial.org/chapters/beginning#fn-1_14) and**bundle install**:

**$** bundle update

**$** bundle install

Fetching source index for https://rubygems.org/

.

.

.

The **bundle install** command might take a few moments, but when it’s done our application will be ready to run.

[1.2.5 rails server](http://ruby.railstutorial.org/chapters/beginning#sec-rails_server)

Thanks to running **rails new** in [Section 1.2.3](http://ruby.railstutorial.org/chapters/beginning#sec-the_first_application) and **bundle install** in [Section 1.2.4](http://ruby.railstutorial.org/chapters/beginning#sec-bundler), we already have an application we can run—but how? Happily, Rails comes with a command-line program, or *script*, that runs a *local* web server, visible only from your development machine:

**$** rails server

=> Booting WEBrick

=> Rails application starting on http://0.0.0.0:3000

=> Call with -d to detach

=> Ctrl-C to shutdown server

(If your system complains about the lack of a JavaScript runtime, visit the [execjs page at GitHub](https://github.com/sstephenson/execjs)for a list of possibilities. I particularly recommend installing [Node.js](http://nodejs.org/).) This tells us that the application is running on [port number](http://en.wikipedia.org/wiki/TCP_and_UDP_port) 3000[15](http://ruby.railstutorial.org/chapters/beginning#fn-1_15) at the address **0.0.0.0**. This address tells the computer to listen on every available IP address configured on that specific machine; in particular, we can view the application using the special address **127.0.0.1**, which is also known as**localhost**. We can see the result of visiting <http://localhost:3000/> in [Figure 1.3](http://ruby.railstutorial.org/chapters/beginning#fig-riding_rails).

[Browsers](http://ruby.railstutorial.org/chapters/beginning#sec-1_2_1_3)

Although there are many web browsers to choose from, the vast majority of Rails programmers use Firefox, Safari, or Chrome when developing. All three browsers include a built-in “Inspect element” feature available by right- (or control-)clicking on any part of the page.

[A note about tools](http://ruby.railstutorial.org/chapters/beginning#sec-1_2_1_4)

In the process of getting your development environment up and running, you may find that you spend a *lot* of time getting everything just right. The learning process for editors and IDEs is particularly long; you can spend weeks on Sublime Text or Vim tutorials alone. If you’re new to this game, I want to assure you that *spending time learning tools is normal*. Everyone goes through it. Sometimes it is frustrating, and it’s easy to get impatient when you have an awesome web app in your head and you *just want to learn Rails already*, but have to spend a week learning some weird ancient Unix editor just to get started. But, as with an apprentice carpenter striving to master the chisel or the [plane](https://en.wikipedia.org/wiki/Block_plane), there is no subsitute for mastering the tools of *your* trade, and in the end the reward is worth the effort.

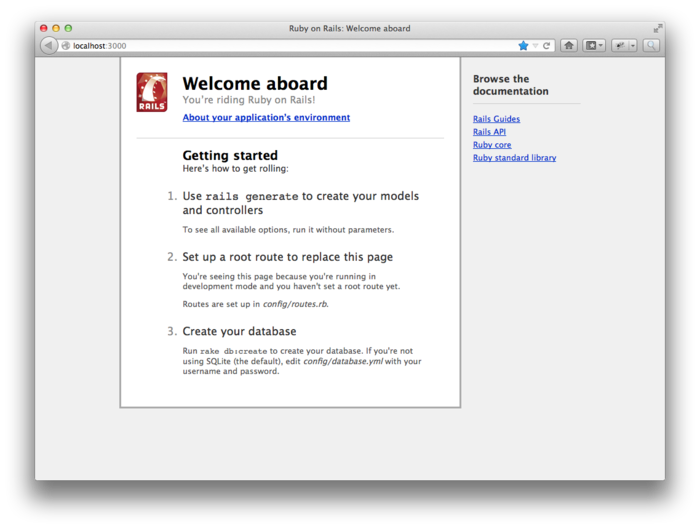


Figure 1.3: The default Rails page. [(full size)](http://railstutorial.org/images/figures/riding_rails_4_0-full.png)

To see information about our first application, click on the link “About your application’s environment”. The result is shown in [Figure 1.4](http://ruby.railstutorial.org/chapters/beginning#fig-riding_rails_environment). ([Figure 1.4](http://ruby.railstutorial.org/chapters/beginning#fig-riding_rails_environment) represents the environment on my machine when I made the screenshot; your results may differ.)

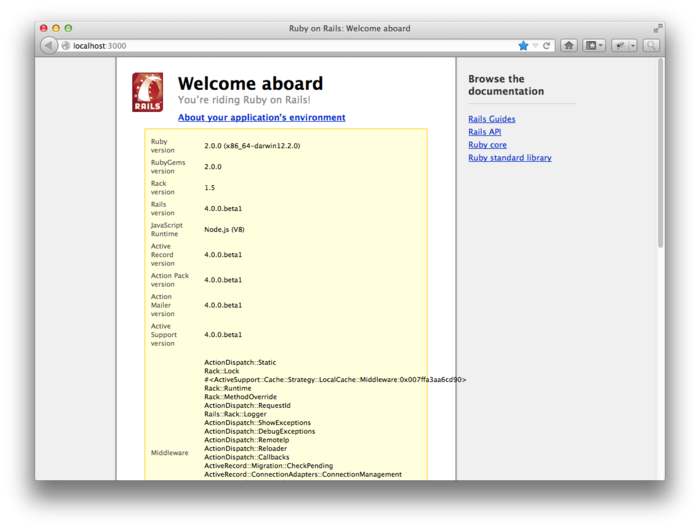


Figure 1.4: The default page with the app environment. [(full size)](http://railstutorial.org/images/figures/riding_rails_environment_4_0-full.png)

Of course, we don’t need the default Rails page in the long run, but it’s nice to see it working for now. We’ll remove the default page (and replace it with a custom home page) in [Section 5.3.2](http://ruby.railstutorial.org/chapters/filling-in-the-layout#sec-rails_routes)