



tag: v2.2.3 tracks / doc / installation.textile

 dnrce on 12 Jul 2014 Add instructions to comment out unused gems2 contributors  

135 lines (73 sloc) 11.833 kb

Raw Blame History  

Installing Tracks

Getting Tracks

There are two methods of downloading Tracks:

1. (Recommended for most people) Download the [zipped package](#) for the latest stable release (2.2) and unzip in your preferred location (e.g. ~/Sites for Mac OS X users).
2. If you want to live on the edge, you can get the latest development version from GitHub using git (bear in mind that this may be less stable than the released versions):

```
cd ~/Sites
git clone git://github.com/tracksapp/tracks.git
cd tracks
```

RootFile with cd as
DeclareReference for
later use by RootFiles in
that directory

Easy installation options

There are a few easy options if you are not confident about installing Tracks from source using these instructions. If you'd like to install Tracks on a local machine, try [BitNami](#) — it runs on Windows, Mac OS X and Linux. But they also support preconfigured virtual machines, including Amazon machine images.

Alternatively, you could try [JumpBox](#), who provide a JumpBox for Tracks. JumpBoxes are pre-built, pre-configured virtual applications which run in a range of [Virtualization software applications](#). You just download the JumpBox (free), then open the file with your Virtualization software. Once the JumpBox has booted, it will give you a URL which you can visit in a browser. The software will then guide you through setting up an account. If you'd like to try out the JumpBox without installing it, you can use the 'Trial This JumpBox' button on the web site, which will let you play around with it to test it out. Furthermore, there is a free public AMI available for Amazon EC2. Just use any EC2 client and search for Tracks. This works in exactly the same way as the downloaded JumpBox you can easily migrate from a downloaded installation to an EC2 instance or back using the backup system of the JumpBox.

Several third parties provide Tracks hosting as a service. A list of these providers can be found [on the wiki](#). Please note that they may run older versions of Tracks.

Requirements

The Tracks interface is accessed through a web browser, so you need to run a webserver to serve the Tracks pages up to you. This isn't as daunting as it sounds, however: Tracks ships with a built-in web server called Mongrel which you can run on your own computer to serve the Tracks application locally. If you want to be able to access Tracks from any computer connected to the Internet, then you need to install Tracks on a publicly accessible server, and you will probably be better off using a more robust web server such as [Apache](#) (using [modrails](#)) or [Lighttpd](#) to serve the pages, particularly if it will be used by many people.

Tracks stores its data in a database, and you can either use SQLite3, MySQL or PostgreSQL. SQLite3 is the best choice for a single user (or a small number of users) on a local installation, while MySQL or PostgreSQL is better for multiple

users on a remote installation.

What is included with the Tracks package?

1. Tracks itself
2. An empty SQLite3 database, set up with the correct database schema

What you need to install

If you don't want to (or can't) use one of the all in one installations, you'll need to install a few things, depending on your platform and your needs.

1. **Ruby.** Tracks requires either Ruby 1.8.7 or Ruby 1.9.x. Ruby 1.9.x is the recommended version. Please note that support for ruby 1.8.7 will be dropped after this release!
2. **RubyGems.** Tracks was tested on version 1.8.24. You may upgrade using `gem update --system`. The gems needed by Rails to interact with the database have to be compiled on the platform on which they will be run, so we cannot include them with the Tracks package, unlike some other gems. So you will need to [download](#) and install RubyGems (run `ruby setup.rb` after extracting the package). If you use Linux, rubygems may be available through your packaging system. Mac OS X users already have RubyGems and the SQLite3 gem already installed on their systems. Once installed you can use RubyGems to install the gems you need for your database. Run `bundle install --without development test` from the directory you installed Tracks in. This will install all needed gems, including those for MySQL and Sqli3. If you do not want one of them, you can comment it out in your `Gemfile` which can be found in the root of the Tracks installation.
3. **Database.** The easiest option is to use SQLite3, as the database is included in the package. You may need to install it first for your platform (see [sqlite.org](#) for downloads and installation instructions). If you want to use MySQL, download and install a package for your platform from [MySQL.com](#). The basic steps for Postgresql should be similar to those for MySQL, but they will not be discussed further here.

Various Tracks users have contributed installation howtos for specific setups. They are [on the wiki](#).

Installation

This description is intended for people installing Tracks from scratch. If you would like to upgrade an existing installation, please see the [upgrade documentation](#).

1. Unzip tracks and install in a directory
2. Decide on a database to use
 - i. SQLite3 – change database.yml to point to SQLite3 database. Make sure you add the complete path to the database. Remove the `mysql2` gem from the Gemfile
 - ii. MySQL – create new MySQL db and grant all privileges
3. Install the necessary prerequisites using Bundler
4. Configure some variables
5. Populate the database with the Tracks schema
6. Start the server
7. Visit Tracks in a browser
8. Customise Tracks

Unzip Tracks and install

Unzip the package and move Tracks into the directory you want to run it from. For example, for Mac OS X users, `~/Sites` is a good choice.

Decide on a database

Before you go any further, you need to decide which database you will use. See the 'What you need to install' section for details on installing the required components for your choice of database.

1. **SQLite3.** All you need to do is make sure that you point Tracks to the included SQLite3 database in `/db` in the next step, 'Configure variables'.

2. **MySQL.** Once you have MySQL installed, you need to create a database and database-user to use with Tracks. For this, you can use MySQL Administrator or go into a terminal and issue the following commands:

```
mysql -uroot -p
mysql> CREATE DATABASE tracks;
mysql> GRANT ALL PRIVILEGES ON tracks.* TO yourmysqluser@localhost \
IDENTIFIED BY 'password-goes-here' WITH GRANT OPTION;
```



RootFile

Install the necessary prerequisites using Bundler

Tracks makes use of several other Ruby libraries (known as 'gems') to provide additional functionality. The Bundler tool makes it easy for the gems that Tracks needs to be installed.

1. Make sure you have bundler on your system already. It can be installed by running `gem install bundler`
2. Edit the file `Gemfile` in the Tracks root directory. You may comment out any database drivers you will not be using. You may also comment out `therubyracer` if you are installing Tracks on Windows or Mac OS X, or if you have another JavaScript runtime such as Node.js installed.
3. Run the command `bundle install --without development,test` in the directory that you unzipped your Tracks download to.
4. Wait for Bundler to finish installing the necessary gems that Tracks needs. This can take some time depending on the speed of your internet connection and the speed of the system you're installing Tracks on.

Configure variables

1. If you downloaded Tracks via GitHub, you need to duplicate the files `database.yml.tpl` and `site.yml.tpl` and remove the `.tpl` extension from the duplicates. Once you've made those copies, edit the files as described in steps 2 and 3.
2. Open the file `/config/database.yml` and edit the `production:` section with the details of your database. If you are using MySQL the `adapter:` line should read `adapter: mysql2`, `host: localhost` (in the majority of cases), and your username and password should match those you assigned when you created the database. If you are using SQLite3, you should have only two lines under the production section: `adapter: sqlite3` and `database: db/tracks-blank.db`.
3. Open the file `/config/site.yml`, and read through the settings to make sure that they suit your setup. In most cases, all you need to change are the `salt: "change-me"` line (change the string "change-me" to some other string of your choice), the administrator email address (`admin_email`), and the time zone setting. For the time zone setting you can use the command `bundle exec rake time:zones:local` to see all available timezones on your machine
4. If you are using Windows, you may need to check the 'shebang' lines (`#!/usr/bin/env ruby`) of the `/public/dispatch.*` files and all the files in the `/script` directory. They are set to `#!/usr/bin/env ruby` by default. This should work for all Unix based setups (Linux or Mac OS X), but Windows users will probably have to change it to something like `#!/c:/ruby/bin/ruby` to point to the Ruby binary on your system.
5. If you intend to deploy Tracks with the built in webserver called WEBrick, you'll need to change `config.serve_static_assets` to `true` in `config/environments/production.rb` in order for the images, stylesheets, and javascript files to be served correctly.

Populate your database with the Tracks schema

Open a terminal and change into the root of your Tracks directory. Enter the following command:

```
bundle exec rake db:migrate RAILS_ENV=production
```



RootFile

This will update your database with the required schema for Tracks. If you are using SQLite3, it is not strictly necessary, because the SQLite3 database included with Tracks already has the schema included in it, but it should not do any harm to run the command (nothing will happen if it is up to date).

Precompile assets

Static assets (images, stylesheets, and javascript) need to be compiled in order for them to work correctly with the new asset pipeline feature in Rails. Precompiling your assets is as simple as running the following command while inside the

Tracks root directory:

```
bundle exec rake assets:precompile
```



RootFile

Start the server

While still in the Terminal inside the Tracks root directory, issue the following command:

```
bundle exec rails server -e production
```



RootFile

If all goes well, you should see some text informing you that the WEBrick server is running: => Rails application starting in production on `http://0.0.0.0:3000`. If you are already running other services on port 3000, you need to select a different port when running the server, using the `-p` option. You can stop the server again by the key combination `Ctrl-C`.

Visit Tracks in a browser

Visit `http://0.0.0.0:3000/signup` in a browser (or whatever URL and port was reported when you started the server in the step above) and chose a user name and password for admin user. Once logged in as admin, you can add other (ordinary level) users. If you need to access Tracks from a mobile/cellular phone browser, visit `http://yourdomain.com/mobile/`. This mobile version is a special, lightweight version of Tracks, designed to use on a mobile browser.

Customise Tracks

Once logged in, add some Contexts and Projects, and then go ahead and add your actions. You might also want to visit the Preferences page to edit various settings to your liking. Have fun!

