Tracks 1.5 Manual

Tracks Development Team

2007-11-25

Revision: $Id: manual.markdown 654 2007-12-16 15:01:05Z bsag $

Formatted for LaTeX
by MultiMarkdown

Contents

CHAPTER

1 Installing Tracks 1.5

1.1 Introduction

Tracks 1.5 has been thoroughly beta tested by a large number of people, and should be fully stable for everyday use. However, once set up, Tracks will contain the majority of your plans for your work and personal life, so it's only sensible to make sure that you have frequent, reliable backups of your data. Full changenotes on the release can be found in `doc/CHANGELOG`. Full API documentation can be found at `doc/app/index.html`, once you have run `rake appdoc`

There are two methods of downloading Tracks 1.5:

1. (Recommended for most people) Download the zipped package, and unzip in your preferred location (e.g. `~/Sites` for Mac OS X users).
2. Download using Subversion:

```
svn co --username=guest
http://www.rousette.org.uk/svn/tracks-repos/tags/current tracks
```

1.2 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 server such as Apache 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.

1.2.1 All-in-one installations

This is the easiest solution for Mac OS X 10.4 or Windows users wanting to run Tracks locally.

1. **Mac OS X**. Locomotive is an all-in one installer for Mac OS X 10.4, which includes everything you need to run Tracks with a SQLite3 database. Locomotive isn't currently Leopard compatible, and doesn't work with Panther.
2. **Windows**. Instant Rails includes everything you need, including MySQL.

1.2.2 What is included with the Tracks package

1. Tracks itself
2. Rails 1.2.5 (installed in the `/vendor/rails` directory, so you do not need to install Rails yourself)
3. An empty SQLite3 database, set up with the correct database schema

1.2.3 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**. Version 1.8.6 is recommended, but it is also possible to use 1.8.5, 1.8.4 and 1.8.2. Note that 1.8.3 is not compatible. If you are running Mac OS X Leopard, you already have Ruby 1.8.6 installed by default, so you have nothing to do here. You can get the source to compile yourself here for all platforms, or Windows users can use an easy installer. If you're using a version of Mac OS X earlier than 10.5.0, it is recommended that you use the instructions here to install all the Rails dependencies, though you can skip the step to install Rails if you like.
2. **RubyGems**. 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). Note that once again, Mac OS X Leopard users get an easy life, because RubyGems and the SQLite3 gem is already installed. Once installed you can use RubyGems to install the gems you need for your database. If you are using SQLite3, run `sudo gem install`

`sqlite3-ruby`, then select the appropriate package for your platform (version 1.2.1 recommended). You can use MySQL without installing a gem, but installing the gem can speed things up a bit: `sudo install gem mysql`. If you're using Leopard, there are a few work-arounds necessary, which are explained on Mac OS Forge. The ruby-mysql bindings can sometimes be a bit troublesome to install, so to be honest, it's probably not worth the bother unless you are trying to wring maximum speed out of your system. If you are using PostgreSQL, then you can install a postgres gem: `gem install postgres`.

3. **Database**. The easiest option is to use SQLite3, as the database is included in the package. All you need then is the `sqlite3-ruby` gem, as described in step 2, and the SQLite3 libraries and binary (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.

1.3 Installation

This description is intended for people installing Tracks from scratch. If you would like to upgrade an existing installation, please see Upgrading to Tracks 1.5 (chapter 2).

1. Unzip tracks (subsection 1.3.1) and install in a directory
2. Decide on a database (subsection 1.3.2) to use
   a) SQLite3 - change database.yml to point to SQLite3 database
   b) MySQL - create new MySQL db and grant all privileges
3. Configure some variables (subsection 1.3.3)
4. Populate the database with the Tracks 1.5 schema (subsection 1.3.4)
5. Start the server (subsection 1.3.5)
6. Visit Tracks in a browser (subsection 1.3.6)
7. Customise Tracks (subsection 1.3.7)

1.3.1 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.

1.3.2 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 (subsection 1.2.3) section for details on installing the required components for you 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 (subsection 1.3.3).

2. **MySQL**. Once you have MySQL installed, you need to create a database to use with Tracks 1.5. Go into a terminal and issue the following commands:

```
mysql -uroot -p
mysql> CREATE DATABASE tracks15;
mysql> GRANT ALL PRIVILEGES ON tracks15.* TO yourmysqluser@localhost \
IDENTIFIED BY 'password-goes-here' WITH GRANT OPTION;
```

1.3.3 Configure variables

1. If you downloaded Tracks 1.5 via Subversion, you need to duplicate the files `database.yml.tmpl` and `environment.yml.tmpl` and remove the `*.tmpl` extension from the duplicates. Similarly, duplicate `/log.tmpl` and remove the `*.tmpl` extension, then 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: mysql`, `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-15.db`.

3. Open the file `/config/environment.rb`, and read through the settings to make sure that they suit your setup. In most cases, all you need to change is the `SALT = "change-me"` line (change the string "change-me" to some other string of your choice), and the time zone setting.

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 *nix 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.

1.3.4 Populate your database with the Tracks 1.5 schema

Open a terminal and change into the root of your Tracks 1.5 directory. Enter the following command:

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

This will update your database with the required schema for Tracks 1.5. 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).

1.3.5 Start the server

While still in the Terminal inside the Tracks 1.5 root directory, issue the following command:

```
script/server -e production
```

If all goes well, you should see some text informing you that the Mongrel server is running: `** Mongrel available at 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.

1.3.6 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.

1.3.7 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!

CHAPTER

2 Upgrading to Tracks 1.5

2.1 Upgrading from Tracks 1.043

This should be a relatively straightforward, and involves the following main steps:

1. Back up (subsection 2.1.1) your existing database and installation of Tracks
2. Install Tracks 1.5 (subsection 2.1.2) in a new directory
3. Copy over (subsection 2.1.3) a few configuration files from your Tracks 1.043 directory. If using SQLite3, copy the old database into the new Tracks 1.5 directory
4. Run `rake db:migrate RAILS_ENV=production` to update your old database (subsection 2.1.4) to the new schema – you did back up your database didn't you?
5. Run `script/server` inside your Tracks 1.5 directory to start up Tracks 1.5 (subsection 2.1.5).
6. Once you are happy that everything is working well, delete your old Tracks directory (subsection 2.1.6).

2.1.1 Backing up

It's very important that you **back up your database** before you start the upgrade process. It's always possible for things to go wrong with the database update, and you don't want to lose any data. If you are using SQLite3 and you are leaving your old Tracks directory in place, then you don't need to do anything. However, there is no harm in taking extra precautions and copying your database from `/db` to a safe location as an extra backup, or making a dump of the schema and contents. You will never regret making too many backups! If you are using MySQL, make a SQL dump of your database, replacing the terms in square brackets with the correct information for your setup:

```
mysqldump ---user [user name] ---password=[password] [database name]
> [dump file]
```

Rename your old Tracks installation (e.g. to 'tracks-old') so that you can install Tracks 1.5 along side it.

2.1.2 Install Tracks 1.5

There are two methods of downloading Tracks 1.5:

1. (Recommended for most people) Download the zipped package, and unzip in your preferred location (e.g. ~/Sites for Mac OS X users).

2. Download using Subversion:

```
svn co --username=guest \
http://www.rousette.org.uk/svn/tracks-repos/tags/current tracks
```

2.1.3 Copy over old configuration files

There are a few files you need to copy over from your old installation. If you copy them over rather than moving them, you can still run your old version of Tracks if anything goes awry with the installation process.

1. Copy /config/database.yml from your old Tracks directory to the same location in the new one. Double check that the information there is still correct.

2. Duplicate /config/environment.rb.tmpl in the Tracks 1.5 directory, and rename the file to environment.rb. Open the file and alter the line SALT = "change-me" so that it matches what you had in this file in your old installation. You may also want to change the time zone setting as appropriate for your location (ENV['TZ'] = 'US/Eastern'). If you have made any other customisations to environment.rb in the past, copy those over, but the contents of the file have changed quite a lot since 1.043, so check it carefully.

3. Copy your /log directory over from your old installation to the root of the new one, or just rename /log.tmpl to log to start afresh.

4. If you are using SQLite3, copy your database from /db in your old Tracks directory to the same location in the new one.

5. If you are using Windows, you may need to check the 'shebang' lines (`#!/usr/bin/env ruby`)[1] of the `/public/dispatch.*` files and all the files in the `/script` directory. They are set to `#!/usr/bin/env ruby` by default. Check the format of those lines in your old installation, and change the new ones as necessary.

2.1.4 Update your old database to the new format

In a terminal, change directories so that you are inside the Tracks 1.5 directory. Then issue the command:

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

Watch the output carefully for errors, but it should report at the end of the process that everything worked OK. If you do get errors, you'll have to fix them before you proceed any further. Running rake with the `--trace` option can help to track down the problem.

2.1.5 Start the server

If you're still in the Tracks 1.5 root directory in a terminal, enter the following command to start up Tracks in production mode:

```
script/server -e production
```

Visit the URL indicated by the output (e.g. `** Mongrel available at 0.0.0.0:3000`) in a browser, and with any luck, you should be able to log in and find all your actions as you left them!

2.1.6 Clean up your old installation

Once you're certain that your new Tracks 1.5 installation is working perfectly, you can delete your old Tracks directory.

2.2 Upgrading from versions prior to 1.043

The best option for versions prior to 1.043 is to follow the instructions below to upgrade to version 1.043, then use the instructions above to upgrade from version 1.043 (section 2.1).

1. For safety, rename your current Tracks directory to 'tracks-old' or something similar.
2. Before you do anything else, **BACK UP YOUR DATABASE** (tables and content) and keep the SQL dumps somewhere safe so that you can recreate the old database if necesary.
3. Download a copy of Tracks 1.043 and unzip alongside your 'tracks-old' directory.

[1] The `env` binary helps to locate other binaries, regardless of their location. If you don't have `env` installed, you'll need to change this line to point to the location of your Ruby binary.The `env` binary helps to locate other binaries, regardless of their location. If you don't have `env` installed, you'll need to change this line to point to the location of your Ruby binary.

4. Open the file `config/environment.rb` and look at the last line which should read: `SALT = "change-me"`. Change the word change-me to something else of your choosing. This string will be used as a 'salt' to encrypt your password and make it a bit more secure. Also look at the timezone setting at the bottom. You can leave it commented out if your server is in the same time zone as you, but you may need to adjust it if your server is in a different time zone.

5. In `database.yml` insert your old database name, user and password under the 'development' section. If you are using SQLite3 rather than MySQL or PostgreSQL, you need only the database name, and to change the 'adapter' line to 'sqlite3'. You also need to copy (NOT MOVE!), your SQLite3 database from your tracks-old `db` directory to your new tracks `db` directory

6. Run the command `rake extract_fixtures` inside the Tracks directory. This will populate the `db/exported_fixtures` directory with `*.yml` files corresponding to the contexts, projects and todos table from the contents of your old database.

7. Open `db/exported_fixtures/todos.yml` and search for the lines starting `created:` and replace with `created_at:`. If you are using SQLite3, you also need to change the following: `done: "0"` with `done: "f"` and `done: "1"` with `done: "t"`. You need to replace the similar 'done' lines in `projects.yml`, and in `contexts.yml` replace `hide: "0"` with `hide: "f"` and `hide: "1"` with `hide: "t"`.

8. Create a new MySQL database (named tracks1043, for example). In `database.yml` insert this new database name, user and password under the 'development' and 'production' sections. If you are using SQLite3, insert a new name for a database to hold your Tracks 1.043 data.

9. Run the command `rake db_schema_import` inside the Tracks directory. This should import the upgraded schema for 1.043 into your new database.

10. Run the command `rake load_exported_fixtures` which will import the contents of your old database from the fixtures files in `db/exported_fixtures`.

11. If you are using Windows, you may need to check the 'shebang' lines (`#!/usr/bin/env ruby`)[2] of the `/public/dispatch.*` files and all the files in the `/script` directory. They are set to `#!/usr/bin/env ruby` by default. Check the format of those lines in your old installation, and change the new ones as necessary.

12. Try starting up the server with `script/server` to make sure that all your data has migrated successfully. If all is well, follow the instructions above to upgrade from version 1.043 (section 2.1) to Tracks 1.5

[2]The `env` binary helps to locate other binaries, regardless of their location. If you don't have `env` installed, you'll need to change this line to point to the location of your Ruby binary.The `env` binary helps to locate other binaries, regardless of their location. If you don't have `env` installed, you'll need to change this line to point to the location of your Ruby binary.