Tracks Manual

Doing Things Properly

Contents

Introduction

Tracks is a web application that is specifically designed to implement the Getting Things Done ™ (GTD) methods. That doesn't mean that you can't use it for other kinds of todo tracking. Data is stored in a database (either MySQL, Postgresql or SQLite), and viewed in a web browser via a web server (Apache, Lighttpd or Mongrel among others). This makes it cross-platform as well as being accessible from anywhere that you have web access.

Using the GTD method as a model, there are three main components to Tracks: Next actions, Contexts and Projects.

- **Next Actions**: These are the heart of GTD. They are the very next physical action that can be taken on something. It's best to phrase these in an active way e.g. "Call Bob about the committee meeting" or "Search for a reputable garage".
- **Contexts**: Contexts are very flexible, and can be places, states of mind or modes of working in which actions can be taken. Next actions can be assigned to and sorted by context so that you know when you are able to make progress with items. e.g. "Library", "Shops" or "Tired".
- **Projects**: any goal which requires more than one next action to take it to completion is a Project. In Tracks, you can view your next actions by Project.

Tracks 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`.

About this manual

The source for this manual is kept in a branch of the main Tracks **GitHub repository**. The source is maintained using **Jekyll** and **Pygments** to create the HTML (which is automatically published as a GitHub page **here** when the repository is pushed), and **Prince XML** is used to convert the HTML to PDF using a rakefile. If you have cloned the source and make changes to the text, you can run delete the existing `manual.pdf` file in the root, then run `rake manual.pdf` to regenerate the PDF file.

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 (2.0) 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):

```
1 cd ~/Sites
2 git clone git://github.com/bsag/tracks.git
3 cd tracks
```

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. 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.

If you'd like an easy way to access Tracks from any internet-connected computer, sign up for a free account at **Morph eXchange**. Sign up for a free account, then click on "See all available subscriptions" and you'll see Tracks listed as an application. Just subscribe to Tracks and you can get started. This option is recommended for those with little technical know-how.

There are also some other free hosted Tracks sites, like http://GTDify.com and http://Tracks.tra.in that you can use.

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** (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. Rails 2.3.11 (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

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 Ruby Version 1.8.7. There are issues when running with Ruby 1.8.6. Also Tracks was not tested on Ruby 1.9.×.
2. **RubyGems**. Tracks was tested on version 1.3.7 and 1.5.0. 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 throught your packaging system. 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 gem install 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.
4. **Rake** You will need rake to install and upgrade Tracks. If it is not installed on your system, use `sudo gem install rake` to install it.

You can find several installation howtos for specific setups **here**. They were contributed by various Tracks users.

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 2.0.

1. Unzip tracks and install in a directory
2. Decide on a database to use
    1. SQLite3 – change database.yml to point to SQLite3 database. Make sure you add the complete path to the database
    2. MySQL – create new MySQL db and grant all privileges
3. Configure some variables
4. Populate the database with the Tracks schema
5. Start the server
6. Visit Tracks in a browser
7. 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 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'.
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:

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

Configure variables

1. If you downloaded Tracks via GitHub, you need to duplicate the files `database.yml.tmpl` and `site.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-20-blank.db`. If you downloaded the zipped file, the database.yml file is already configured to use the provided SQLite3 file.

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 is the `salt: "change-me"` line (change the string "change-me" to some other string of your choice), and the time zone setting. For the time zone setting you can use the command `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.

Populate your database with the Tracks schema

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

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

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).

Start the server

While still in the Terminal inside the Tracks 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.

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!

Upgrading Tracks

Upgrading from Tracks 2.0RC2 to 2.0

The upgrade to the latest stable 2.0 release consists of the same steps as upgrading from 1.7.x in the following paragraph, just replace 2.0RC with 2.0 where appropriate

PLEASE NOTE that the requirements for Tracks have changes:

1. Rubygems needs at least version 1.3.7 or above. 1.3.7 and 1.5.0 are tested. 1.7.0 does not work for this release of Tracks
2. Only Ruby 1.8.7 is supported, both 1.8.6 and 1.9.x are not tested and are known to have issues.

Upgrading from Tracks 1.7.x or from 2.0RC1 to Tracks 2.0 final

NOTE

1. these instructions will work too if you are upgrading to latest 2.0devel tree.
2. Actually no 2.0RC1 release was done, but some people have been using a development version referred to as RC1. Upgrading is the same as from 1.7.x to RC2.

To upgrade:

1. Back up your existing database and installation of Tracks
2. Install Tracks 2.0RC in a new directory
3. Copy over the configuration from your previous Tracks installation. If using SQLite3, copy the old database into the new Tracks 2.0RC directory.
4. Run `rake db:migrate RAILS_ENV=production` to update your old database to the new schema — you did back up your database didn't you?
5. Run `script/server` inside your Tracks 2.0RC directory to start up Tracks 2.0RC2.
6. Once you are happy that everything is working well, delete your old Tracks directory.

If you did not start with a new copy of Tracks as described above, but installed over an older version (or used git to pull in a newer version), you need to remove the cached version of the javascript and stylesheets of Tracks.

1. jquery-cached.js from public/javascripts
2. tracks-cached.js from public/javascripts
3. trachs-cached.css from public/stylesheets

If you are running an older version of Tracks (1.8devel), they could also be called jquery-all.js, tracks.js and all.css

Upgrading from Tracks 1.7 or 1.7.1 or 1.7.2 to 1.7.3

The 1.7.1, 1.7.2 and 1.7.3 releases are bug fix releases that do not contain changes to the database structure. You can unzip the new release over your current 1.7 or 1.7.1 install.
It is also possible to follow the instructions below and copy your existing site.yml and database.yml from your Tracks 1.7 install.

Upgrading from Tracks 1.7RC2 to Tracks 1.7

There were only a few fixes in the code, so you can consider unzipping the new release over your install of Tracks 1.7RC2. Do make a backup of your Tracks 1.7RC2 install and your database!

It is also possible to follow the instructions below and copy your existing Site.yml from your Tracks 1.7RC2 install.

Upgrading from Tracks 1.5, 1.6 or from Tracks 1.7RC1 to Tracks 1.7

In Tracks 1.7 (since RC2) the site specific configuration is moved from `environment.rb` into the new `site.yml`. This makes updating `environment.rb` much easier without you needing to set your site specific settings after each update.

After you install Tracks 1.7 there will be no `environment.rb.tmpl` anymore. You will find an `environment.rb` which you can leave untouched. Just fill in your settings from your old `environment.rb` in the new `site.yml`. If you have made any other customisations to `environment.rb` in the past, you can put them in your own configuration file (for example, in my-config.rb) in `config/initializers`. Please let us know it you think they should be in `site.yml.tmpl`

Also, there were some database changes made in Tracks 1.7, so you need to migrate to them.

1. Back up your existing database and installation of Tracks
2. Install Tracks 1.7 in a new directory
3. Copy over the configuration from your previous Tracks installation (except for environment.rb, see above). If using SQLite3, copy the old database into the new Tracks 1.7 directory.
4. Run `rake db:migrate RAILS_ENV=production` to update your old database to the new schema — you did back up your database didn't you?
5. Run `script/server` inside your Tracks 1.7 directory to start up Tracks 1.7.
6. Once you are happy that everything is working well, delete your old Tracks directory.

Upgrading from Tracks 1.043 to 1.7

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

1. Back up your existing database and installation of Tracks
2. Install Tracks 1.6 in a new directory

3. Copy over the configuration from your previous Tracks 1.043 installation. If using SQLite3, copy the old database into the new Tracks 1.7 directory.
4. Run `rake db:migrate RAILS_ENV=production` to update your old database to the new schema — you did back up your database didn't you?
5. Run `script/server` inside your Tracks 1.7 directory to start up Tracks 1.7.
6. Once you are happy that everything is working well, delete your old Tracks directory.

Detailed steps

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:

```
1 mysqldump –-user [user name] –-password=[password] \
2 [database name] > [dump file]
```

Rename your old Tracks installation (e.g. to 'tracks-old') so that you can install Tracks 1.7 along side it.

Install the latest version of Tracks

There are two methods of downloading Tracks:

1. (Recommended for most people) Download the **zipped package**, 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):

```
1 cd ~/Sites
2 git clone git://github.com/bsag/tracks.git
3 cd tracks
```

Copy over old configuration

There are a few settings and configuration 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/site.yml.tmpl` in the Tracks 1.7 directory, and rename the file to `site.yml`. Open the file and alter the line `salt: "change-me"` so that it matches what you had in the `environment.rb` file in your old installation. You may also want to change the time zone setting as appropriate for your location. If you have made any other customisations to `environment.rb` in the past, you can put them in your own configuration file (e.g. my-config.rb) in `config/initializers`. Please let us know it you think they should be in `site.yml.tmpl`.
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.

Update your old database to the new format

In a terminal, change directories so that you are inside the Tracks 1.7 directory. Then issue the command to update your Tracks 1.6 database to the format required for Tracks 1.7:

```
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.

Start the server

If you're still in the Tracks 1.7 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! 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.

Clean up your old installation

Once you're certain that your new Tracks 1.7 installation is working perfectly, you can delete your old Tracks directory.

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.

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.
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`) [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.
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 to Tracks 1.6. 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.

[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.