# Trapper Documentation

***Release 1***

**Trapper development team**

December 16, 2013

# Contents

**Welcome stranger!**

This document is dedicated to help you get Trapper up and running on your local machine or a dedicated server. If you are a user of one of Trapper's instances, you might benefit from the tutorials section as well.

CHAPTER 1

# Documentation contents

## 1.1 Installation guide

The project is developed using:

- Python 2.7.x
- Django 1.6

System requirements:

- virtualenv
- RabbitMQ server
- PostgreSQL 9.1 (tested on version 9.1.9)
- Geospatial libraries: GEOS and PROJ.4
- PostGIS (preferably 2.0.x)

### 1.1.1 System requirements

Let us start with the installation of the system dependencies. Django and other python requirements will be installed locally inside the project using virtualenv.

**Warning:** This guide will cover the installation on the Ubuntu/Debian systems. For other systems, please use the recommended methods of package installation.

Command below will install virtualenv, postgresql, geospatial libraries, PostGIS and rabbitmq-server:

```
user@home:~$ sudo apt-get install python-virtualenv postgresql-9.1 binutils
libproj-dev gdal-bin postgresql-server-dev-9.1
rabbitmq-server libavcodec-dev libavformat-dev libswscale-dev libjpeg-dev python-dev
```

Before installing postgis, check which version is available at your system:

```
user@home:~$ sudo apt-get install -s postgresql-9.1-postgis | grep postgis
```

It may be the case that your version is 1.5.x or lower.

You can either install the available version same as before:

```
user@home:~$ sudo apt-get install postgresql-9.1-postgis
```

Or you could try to find the way to obtain PostGIS 2.x for your system:

**See Also:**

- Adding UbuntuGIS (contains PostGIS 2.0)
- PostGIS 2.1 on Debian Wheezy

**See Also:**

- Installing geospatial libraries
- Installing PostGIS

### 1.1.2 Preparing PostgreSQL database

In this section we will prepare the database for handling the geospatial data, as well as setup some basic user privileges. Before we downloading Trapper perform the following actions in a console:

Now we will create a new postgresql user named *trapper*.

```
user@home:~$ adduser trapper
(use unix defaults here)
user@home:~$ su - postgres
postgres@home:~$ psql
postgres=# CREATE USER trapper WITH PASSWORD 'trapper';
CREATE ROLE
postgres=# \q
```

Snipplet above does the following:

1. Creates a new unix user called *trapper*
2. Switches to *postgres* user
3. Runs *psql* command-line interface and connects to a *template1* database
4. Creates the postgresql user *trapper*
5. `\q` exists the psql

**See Also:**

- Adding postgresql user accounts

**Note:** In production code you will most likely want to create a user with a more sophisticated password (don't forget to update the `trapper/settings.py` accordingly).

**Note:** In order to run trapper's automated unit tests user might need some additional privileges for creating databases. In order to resolve that without assigning him superuser provileges, see Obtaining sufficient privileges section of the Django documentation.

Now we will create a database and set up the postgis extension on it:

```
user@home:~$ su - postgres
postgres@home:~$ psql
postgres=# CREATE DATABASE trapper_db OWNER trapper;
CREATE DATABASE
postgres=# GRANT ALL PRIVILEGES ON DATABASE trapper_db TO trapper;
GRANT
postgres=# \c trapper_db
You are now connected to database "trapper_db" as user "postgres"
trapper_db=# CREATE EXTENSION postgis;
CREATE EXTENSION
trapper_db=# \q
```

Snipplet above does the following:

1. Switches to a *postgres* user
2. Runs *psql* command-line interface and connects to a *template1* database
3. Creates the database named *trapper_db* and makes *trapper* its owner
4. Connects to the newly created database
5. installs the postgis extension (required for the geospatial features of Trapper)
6. `\q` exists the psql

**Warning:** Creating the postgis extension (last step of the snipplet above) is a little bit more involved with PostGIS 1.x. For more details on how to set up a PostGIS 1.x database please refer to:

- Setting up PostGIS 1.x (Django documentation)

### 1.1.3 Preparing the project

Next step is cloning the repository and installing python the requirements.

```
user@home:~$ git clone https://github.com/TrapperTeam/Trapper
user@home:~$ cd Trapper/
user@home:~$ virtualenv env
user@home:~$ ./env/bin/pip install -r requirements.txt
```

### 1.1.4 Running Trapper

The project is now set up and ready to use. Initialize the database along with the test data and run the server:

```
user@home:~$ ./setup_database.sh
user@home:~$ ./run_server.sh
```

Additionally, execute a celery worker in a separate shell:

```
user@home:~$ ./run_celery.sh
```

### 1.1.5 Extra: Generating this documentation

Since Trapper is developed using virtualenv, it may be difficult to generate this documentation using sphinx-build that's installed system-wide. In order to resolve that simply perform the following steps:

**Note:** `$TRAPPER_ROOT` is the path the Trapper on your system, e.g. `/home/user/MyProjects/Trapper/`

```
user@home:~$ cd $TRAPPER_ROOT
user@home:~$ which sphinx-build
/usr/bin/sphinx-build (or other system path)
user@home:~$ source ./env/bin/activate
user@home:~$ which sphinx-build
$TRAPPER_ROOT/env/bin/sphinx-build
```

After that you should be able to generate this document without problems:

```
user@home:~$ cd $TRAPPER_ROOT/docs/
user@home:~$ make html
```

After that, this website will reside in `$TRAPPER_ROOT/docs/_build` directory.

## 1.2 Tutorials

These tutorial will cover some of the actions you can perform in Trapper. Trapper allows different features depending on your role and participation in media classification projects. This tutorial page will be roughly divided according to these differences

### 1.2.1 System roles

Roles in Trapper are hierarchically ordered. The following roles exist within the Trapper:

- *Anonymous user*
- *Crowd-Sourcing user*
- *Expert user*
- *Trapper staff*

The least privileged type of user is the `Anonymous` one. Each successive role introduces more permissions:

**Note:** The hierarchy of *privileges* is as follows:

`Staff` > `Expert` > `Crowd-Sourcing` > `Anonymous`

Each role on the **right** of any other role has **less** privileges and features. At the same time, each role on the **left** has all the features and privileges as your role.

E.g.: if your role in the Trapper is `Expert`, tutorials for roles `Crowd-Sourcing` and `Anonymous` will also be relevant you.

### 1.2.2 Anonymous user

Anonymous users of Trapper have a limited functionality. Namely, they cannot alter Trapper's data in any way nor contribute to any of the ongoing media classification projects.

**Note:** As an anonymous user you have access to the following modules of Trapper:

- Home page

- List of publicly available resources, collections and locations
- List of projects by their name

### Possible actions

- List of publicly available resources `Storage > Resources`
- List of publicly available collections `Storage > Collections`
- List of publicly available locations (displayed on the map) `Geomap > Show map`
- You can see a list of project by their names but you can't see the details on any of it `Media classification > Projects`

## 1.2.3 Crowd-Sourcing user

**Note:** Once you log-in to Trapper, are recognized as a `Crowd-Sourcing` user, navigation bar will display more navigation options, namely:

- Profile view
- Messaging features
- Crowd-Sourcing menu under `Media Classification` drop-down

### Edit Profile

Since you are a registered user now, you can edit your user profile, to do so navigate to your profile view (your user name next to a `Logout` button on the right).

When you click it, you can see the details about your profile. Below the table, you should see a button called `Edit Profile`. Clicking it will take you to the profile editing view in which you can alter the information about yourself.

### Classifying material

In order to classify media resource go to navigation bar and select `Media classification > Classify material`.

At the moment, Trapper lets you pick the crowd-sourcing enabled resources for the classification. On the newly opened page you will see a list of ongoing media classification projects, and the resources available for the classification. Once you select given resource, you will see the media file associated with it (usually an image or a video). Your task is to describe the situation visible on the media according to some `Features`, e.g. How many animals are visible, and of what kind.

Once you make your decision, you can submit your classification data to the system.

### Send and receive messages with other users

Registered users have the ability to use Trappers messaging module.

**Note:** The module itself tries to mimic a standard (but very limited) **e-mail** functionality:

- Each message can be directed towards a single users
- Each message is composed of a *subject* and a *body*

In order to send a message to another user, navigate to `Messaging > New Message` at the main navigation bar. You will be presented with a simple message form, in which you are to enter the message subject, the message itself, and a recipient.

Additionally to writing new messages, you can browse and read the incoming and outgoing correspondence by navigating to `Messaging > Inbox` and `Messaging > Outbox` correspondingly.

**Note:** Unread incoming messages and system notifications are also indicated by the altered icon of the `Messaging` menu at the main navigation bar.

### 1.2.4 Expert user

Expert users are usually researchers operating on the whole collections of resources along with the context and the classification results. In future iterations of Trapper, this role will have access to many analytical modules, such as generating statistics and modeling patterns of behaviour of animals within the geo-spatial context.

**Note:** `Expert` users can benefit from many more of Trapper's features:

- Create new `Media Classification` projects
- Upload resources, define collections, upload and define locations and request

### 1.2.5 Trapper staff

Trapper's staff members are usually the system administrators and the core maintainers of the Trapper instance. They have all the privileges by default, and have an access to *Django*'s admin panel.

**Note:** Staff users will have an additional `Admin Site` button at the main navigation bar which takes them to the Django's admin panel.

## 1.3 Technical documentation: Overview

This document will cover a technical documentation for the Trapper project.

### 1.3.1 Introduction

Trapper was developed with a flexible architecture in mind. For that reason it was separated into few django applications which establish some layers of abstraction over common usages. As of yet, there exist few core trapper applications, which often communicate between each other.

### 1.3.2 Django applications

Trapper is composed of several Django applications. In next section we will cover each application in the project, describing the *Models* (database ORM definition) of each application, its *Views* (*Controller* in the MVC) as well as the frontend, i.e. *Templates*. In many cases we will also cover the *Forms* or the *Decorators* of the application.

Core applications of Trapper are following:

- Accounts (*accounts*)
- Storage (*storage*)
- Media classification (*media_classification*)
- Messaging (*messaging*)
- Geomap (*geomap*)
- Common (*common*)

### 1.3.3 Coding style

#### General rules

- We aim to follow PEP-8 coding style guide, although at this stage of development it wasn't as carefully abiden.
- Exception to the rule above is using **tabs** instead of spaces for indentation, and allowing for a line longer than 79 characters.

#### Strings: single vs double quotes

Use double quotes:

- When the string is a sentence, message or a word: `"Four Torkshiremen."`, `"Shoebox"`

Use single quotes:

- When the string is a regexp: `r'spam/detail/(?P<pk>\d+)/$'`
- When the string is an identifier, kwarg which is not a sentence or a dict key:

```
FRUIT_CHOICES = {
    'A': "Apples",
    'B': "Banana",
}

def set_color(color='red'):
    """Sets the color."""
```

#### Intentation in templates

- Python code in templates should be indented up to the current indentation level of the preceeding HTML tag
- Each variable should be separated by one space from the parenthesis, e.g. `{{ value }}` and **not** `{{value}}`

```
<table>
{% for row in table_rows %}
    <tr>
    {% for cell in row.cells %}
```

```
        <td>{{ cell }}</td>
    {% endfor %}
    </tr>
{% endfor %}
</table>
```

## 1.4 Technical documentation: Core modules

### 1.4.1 accounts

Accounts module is responsible for extending the `django.contrib.auth.models.User` model. It provides a `trapper.apps.accounts.models.UserProfile` model which extends some basic user functionality.

#### models

#### views

#### forms

### 1.4.2 storage

Purpose of storage module is uploading and storing various media resources. In most cases this will be video files, images or audio files, but any type of file can be uploaded and stored. It is often used by other applications (e.g. `trapper.apps.media_classification`) to display media resources.

#### models

#### views

#### forms

### 1.4.3 media_classification

Media classification module is at the moment one of the core features of Trapper. This module allows for the classification of media resources (*storage*) within the context defined in `trapper.apps.media_classification.models.Project` objects.

**Note:** It is worth noting how the classification is handled in the backend: Single `Classification` is split into rows, each containing an indentical set of features. Each `ClassificationRow` is again split into a set of *states* of the Feature model, namely an instance of model `FeatureAnswer`

See models for more details:

- `trapper.apps.media_classification.models.Classification`
- `trapper.apps.media_classification.models.ClassificationRow`
- `trapper.apps.media_classification.models.FeatureAnswer`

#### models

#### views

#### forms

#### decorators

### 1.4.4 messaging

Module *messaging* serves as a simple notification system as well as an internal *e-mail* functionality between the users.

#### models

#### views

#### forms

### 1.4.5 geomap

Module *geomap* uses geospatial information about resources stored by *storage* to present it on the map.

#### models

#### views

#### forms

### 1.4.6 common

Module *common* servers as a place for a general-purpose functions or decorators, not associated with any of the applications above.

#### decorators

## 1.5 Troubleshooting

This section will cover some of the problems that you might encounter when trying to setup Trapper for the first time. We rely on many external libraries, and there's no way we could cover the step-by-step installation on every possible system and configuration.

Below you will find a list of problems:

### 1.5.1 Cannot determine PostGIS version

**Warning:** django.core.exceptions.ImproperlyConfigured: Cannot determine PostGIS version for database "trapper_db". GeoDjango requires at least PostGIS version 1.3. Was the database created from a spatial database template?

There may be two problems here:

1. You did not install the PostGIS extension on the database
2. Your PostGIS version is neither 1.3, 1.4, 1.5 or 2.0

Let us check whether you have PostGIS installed. Execute these commands:

```
su - postgres
psql trapper_db
trapper_db=# select postgis_lib_version();
```

1. If the output is an error, you have not installed the PostGIS extension on your database.
2. If the output is silent (e.g. no output), you might have a PostGIS 2.1 or newer which is not recognized properly by GeoDjango.

In order to resolve that, you can edit the Trapper's `settings.py` file and add the following:

`POSTGIS_VERSION = (2, 1, 0)`, where the tuple `(2, 1, 0)` will be your version of PostGIS.

### 1.5.2 OperationalError: FATAL: Peer authentication failed

**Warning:** OperationalError: FATAL: Peer authentication failed for user "trapper"

This error means that user `trapper` cannot access the database.

Find the following line in the file `/etc/postgresql/9.1/main/pg_hba.conf`:

```
# "local" is for Unix domain socket connections only
local       all        all            peer
```

and change the last item, `peer`, to `md5`.

```
# "local" is for Unix domain socket connections only
local       all        all            md5
```

Restart the postgresql server as root:

```
sudo /etc/init.d/postgresql restart
```

## 1.6 Contribute to Trapper

If you would like to contribute to the development of Trapper, or if you have ideas for features, feel free to contact us on IRC: `irc.freenode.net` on channel `#trapper`

### 1.6.1 IRC Webchat

## 1.7 Authors

Trapper was designed and created by the researchers at the Mammal Research Institute of the Polish Academy of Sciences

### 1.7.1 Trapper development team

| Contributor | Role | Contact |
|---|---|---|
| Krzysztof Nowak | Programming | kiryx7@gmail.com |
| Jakub Bubnicki | Programming, Concept | kbubnicki@ibs.bialowieza.pl |
| Marcin Churski | Concept | |
| Leonardo Andreade | Programming, Concept | |
| Dreis Kuijper | Concept | |

CHAPTER 2

# Indices and tables

- *genindex*
- *modindex*
- *search*