Chimère Documentation

Release 1.2

Étienne Loks

October 07, 2012

CONTENTS

CHAPTER

ONE

CHIMÈRE INSTALLATION

Author Étienne Loks

date 2012-02-15

This documents presents the installation of Chimère for version 1.x. This documentation is no longer relevant for version 2.

1.1 Base installation

1.1.1 Installation

Prerequisites

- apache version 2.x with mod_python 3.x
- python versions 2.6 or 2.7
- geodjango version 1.0
- postgres version 8.x
- gettext
- psycopg2
- Python Imaging Library
- Beautiful Soup
- lxml

geodjango is a part of django version 1.0 but it has some specific (geographically related) additionnal dependencies:

- geos 3.0.x
- proj.4 4.4 to 4.6
- posgis versions 1.2.1 or 1.3.x
- gdal

Optionaly (but recommanded):

- tinymce

- gpsbabel

The simpliest way to obtain these packages is to get them from your favorite Linux distribution repositories (for instance python, python-django, python-beautifulsoup, tinymce, apache2, libgeos-3.2.0, proj, gdal-bin, python-gdal, python-lxml, python-psycopg2, python-imaging, gettext, postgresql-8.4 and postgresql-8.4-postgis packages for Debian Squeeze). If these packages do not exist in your distribution's repository, please refer to the applications' websites.

Database configuration

Now that postgres and postgis are installed, you need to create a new user for chimere:

```
createuser --echo --adduser --createdb --encrypted --pwprompt chimere-user
```

Then, you have to create the database, and initialize the geographic types (adapt the paths accordingly to your needs):

```
createdb --echo --owner chimere-user --encoding UNICODE chimere "My Chimère database"
createlang plpgsql chimere
psql -d chimere -f /usr/share/postgresql/8.4/contrib/postgis-1.5/postgis.sql
psql -d chimere -f /usr/share/postgresql/8.4/contrib/postgis-1.5/spatial_ref_sys.sql
```

Getting the sources

The last "stable" version is available in this directory.

Another solution is to get the last git version:

```
git clone git://www.peacefrogs.net/git/chimere
git tag -l # list tagged versions
git checkout v1.2.2 # checkout the desired version
```

Install the sources

Unpack and move the files in an apache user (www-data for Debian) readable directory:

```
sudo mkdir /var/local/django
cd /var/local/django
sudo tar xvjf /home/etienne/chimere-last.tar.bz2
sudo chown -R etienne:www-data chimere
```

In your chimere application directory create settings.py to fit to your configuration. A base template is provided (settings.py.example):

```
cd chimere/chimere/
cp settings.py.example settings.py
vim settings.py
####
PROJECT_NAME = u'Chimère'

ROOT_PATH = '/var/local/django/chimere/chimere/' # path to the installation of Chimère

SERVER_URL = "http://www.peacefrogs.net/" # root of the web address of Chimère
EXTRA_URL = 'chimere/' # suffix to the web address of Chimère
BASE_URL = SERVER_URL + EXTRA_URL
EMAIL_HOST = 'localhost' # smtp of an email server to send emails

TINYMCE_URL = SERVER_URL + 'tinymce/'
```

```
# chimere specific
DEFAULT_CENTER = (-1.679444, 48.114722) # default center of the map
EPSG_PROJECTION = 900913 # projection used for data exchange (JSON flow)
EPSG_DISPLAY_PROJECTION = 4326 # projection used to display on the map

# default id category to check on the map
DEFAULT_CATEGORIES = [1] # list of default category ids checked on the map

# JS definition of the main map cf. OpenLayers documentation for more details
# to begin you can leave the default OpenStreetMap map rendered with Mapnik
MAP_LAYER = "new OpenLayers.Layer.OSM.Mapnik('Mapnik')"

# setting the appropriate language code for your site
LANGUAGE_CODE = 'en-gb'

# database configuration
DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'chimere',
        'USER': 'chimere-user',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
    },
}
```

If you want to use tinymce don't forget to make it available to Chimère.

In this same chimere directory, make a symbolic link to django's basic styles (do not forget to change the path according to your configuration, it is the last time I will recall it to you. Next time, you are on your own!):

```
ln -s /usr/share/pyshared/django/contrib/admin/media/ .
```

Compiling languages

If your language is available in the locale directory of chimere, you will just need to get it compiled. Still being in the chimere directory, this can be done with (here, "de" stands for german. Replace it with the appropriate language code):

```
django-admin compilemessages -l de
```

If your language is not available, feel free to create the default po files and to submit it, contributions are well appreciated. Procedure is as follows :

You first need to create the default po file (of course, replace "de" according to the language you chose to create):

```
django-admin makemessages -l de
```

There should now be a django.po file in locale/de/LC_MESSAGES. Complete it with your translation.

Now that the translation file is completed, just compile it the same way you would have if the language file was already available.

Database initialisation

Create the appropriate tables (still being in chimère application directory):

```
./manage.py syncdb
```

You will be prompted for the creation of an administrator account (administration can be found at: http://where_is_chimere/admin). The database is set, congratulations!

Webserver configuration

Apache configuration with mod_wsgi

Install mod_wsgi for apache:

```
sudo apt-get install libapache2-mod-wsgi
```

Create and edit a configuration for Chimère:

```
sudo mkdir /var/local/django/chimere/apache
sudo cp /var/local/django/chimere/docs/conf/django.wsgi /var/local/django/chimere/apache/django.wsgi
sudo cp /var/local/django/chimere/docs/conf/apache-wsgi.conf /etc/apache2/sites-available/chimere
```

Adapt the files django.wsgi (with the correct sys path) and chimere.

To activate the website reload apache:

```
sudo a2ensite chimere
sudo /etc/init.d/apache2 reload
```

Apache configuration with mod_python

Install mod_python for apache:

```
apt-get install libapache2-mod-python
```

Create and edit a configuration file for Chimère:

```
sudo vim /etc/apache2/sites-available/chimere
```

Insert Apache directives for your installation:

```
# part of the address after the root of your site
<Location "/chimere/">
# directory path to the father of the installation of Chimère
PythonPath "['/var/local/django/'] + sys.path"
SetHandler python-program
PythonHandler django.core.handlers.modpython
SetEnv DJANGO_SETTINGS_MODULE chimere.settings
# set it to on or off if in test or production environment
PythonDebug On
# put differents interpreter names if you deploy several Chimère
PythonInterpreter chimere
</Location>
```

To activate the website reload apache:

```
sudo a2ensite chimere
sudo /etc/init.d/apache2 reload
```

Now that you have gone through ALL this configuration procedure (which was not that hard after all) you can configure the site.

1.2 Base configuration

When you have installed the application there is a few simple steps to follow to configure *your* Chimère.

Most of theses steps are done in the administration pages accessible at : http://where_is_chimere/admin To access theses pages you have to identify you with login and password provided at the initialization of the database.

1.2.1 Creating users

If you are not the only administrator of this Chimère installation you have to create account for the other users. Currently the process has to be done manualy.

Simply click on the Add button near Users. Give a name and a default password (the user can change it on in own later). Then complete the other pieces of information. Check the case: Member of the staff (or this user will not be able to log to this administration site). To simply give this user correct rights don't add permission manualy but make this user member of a group. Two default group are proposed: application administrator and moderator.

Detail of rights for default user/groups:

Task	Application owner	Application administrator	Moderator
User add/modify/delete	yes	no	no
Group add/modify/delete	yes	no	no
Property model add/modify/delete	yes	no	no
News add/modify/delete	yes	yes	no
Area add/modify/delete	yes	yes	no
Icon add/modify/delete	yes	yes	no
Category/Subcategory add/modify/delete	yes	yes	no
Point Of Interest add/modify/delete	yes	yes	yes
Route add/modify/delete	yes	yes	yes

1.2.2 Setting the welcome page

The message has to be set by updating the file templates/welcome.html. You only have to change the message at the begin of #detail_content.

1.2.3 Creating property models

A basic installation of Chimère only permit to associate a name, a category and (for the point of interest) a picture for each point of interest and each route. You may want to add more fields like phone number or opening hours. For that all you have to do is to add a new property model. The administration ask you for name, order (to order between other properties), availability for the user and type (only text field and long text field are available for the moment). Then to make this property available it is necessary to restart your application (and then probably to reload Apache). All forms are then automaticaly updated with this new field.

1.2.4 Updating the detail window

When clicking on a POI a window appear with the details. Particulary if you have set some new property models you may want to customize this window. Each property is in a paragraph with id: property_i_j (i is the order and j is the model property id - the first model property is id 1 then 2...). You can simply adapt the CSS file (static/styles.css) to match the desired presentation. If you want to really change the whole presentation you can change the template file templates/detail.html (go to the Django template documentation for details).

1.2.5 Updating the design

You can of course customize Chimère with your own CSS. To do that just edit the file static/styles.css.

After this basic configuration done the harder is done. You can do now application administration tasks.

1.3 Site administration

The explanation are to create new elements. To modify these elements it is the same if only some fields are already filled.

1.3.1 Creating news

A news system is available. All you have to to do is to click on the Add button near News. For each news you have to provided a name and a content. The content can contain HTML tags. The avaibility is set with a checkbox.

1.3.2 Creating categories/subcategories

Before adding categories you have to set some icons. Theses icons appears on the map and in the categories' box on the main map. Be careful to resize correctly your icons. Indeed the icon will be presented at their real size on the map. To add icons: the Add button near Icons.

Categories are in fact only containers for subcategories. You'll have to provide only a name and an order. To add categories: the Add button near categories (quite clear now, isn't it?)

Fields of subcategories are: an associated category, a name, an icon, an order, a color and an element type. Theses fields are mainly quite self-explainatory. The color is used to draw routes (if this subcategory contains routes). If it a basic color it can be set with the english name (for instance: "red", "blue", "yellow" or "purple") otherwise you can put the HTML RVB code (for instance "#9227c9"). The element type is the type of element the subcategory can contain: POI, route or both.

1.3.3 Creating areas

Areas are useful to provide a quick access to a particular town, a district, etc. To define an area fill a name and move/zoom to the choosed location. Submit it and that's all.

1.4 Moderation

The moderation step is quite simple. It works the same with POIs and routes. The moderator can access to all POIs (or routes) by clicking on the Modify button. A search field is available to search by name but the more interesting is to filter POIs (or route) by state and by subcategory. Then to modify an item you have to click on his name. The submission can now freely modified. Compared to the main submission interface there is only on field add: the state field. To be publish in the main site the item must have the state: Available. If an item is not revelant the Delete button permit to remove it.

CHAPTER

TWO

CHIMÈRE UPGRADE

Author Étienne Loks

date 2012-02-15

This documents presents the upgrade of Chimère to version 1.2. This documentation is no longer relevant for version 2.

2.1 Get new version of dependencies

2.1.1 From version prior to 1.1 to 1.1

Upgrade Django to the 1.2 version. Install the Beautiful Soup library.

2.2 Get the new version

First of all get the new version of the code source.

2.2.1 Download archive from the download site

Versions are available at this address. Take care of getting the last version in the desired X.Y branch (for instance the last version for the 1.0 branch is version 1.0.2 as the time of writing of this document). Extract it to the desired destination path.:

```
su
wget http://www.peacefrogs.net/download/chimere -q -O -| html2text
(...)
[[   ]] chimere-1.0.0.tar.bz2     17-Nov-2010 16:51  53K
[[   ]] chimere-1.0.1.tar.bz2     17-Nov-2010 16:51  53K
[[   ]] chimere-1.0.2.tar.bz2     17-Nov-2010 16:51  53K
(...)

wget http://www.peacefrogs.net/download/chimere/chimere-1.0.2.tar.bz2
mv chimere-1.0.2.tar.bz2 /var/local/django
cd /var/local/django
tar xvjf chimere-1.0.2.tar.bz2
cd chimere-1.0.2
```

2.2.2 Get from the Git repository

Clone the Git repository, checkout the desired version and copy it to the desired destination path.:

```
git clone git://www.peacefrogs.net/git/chimere
cd chimere
git tag -l
(...)
v1.2.0
v1.2.1
git checkout v1.2.1
cd ..
mv chimere /var/local/dgango/chimere-1.0
cd /var/local/dgango/chimere-1.0
```

2.3 Copy files from your old installation

From your old installation at least copy "settings.py" and the content of "static/icons/" and "static/upload/" to the new installation. You have probably customised some styles and templates (for instance "styles.css", "welcome.html" and "base_user.html") don't forget to copy them and eventualy adapt them (if you have old vanilla version of this file comparing with the new one provided is easier).

2.4 Adapt settings.py

The format of settings.py could have evolved, the easiest way to complete your settings.py is to compare your old settings.py.example and the new one provided.

2.4.1 Specific to upgrade from version 1.0 to version 1.1

Version 1.1 of Chimère uses Django 1.2 and with it the manner to define database has changed.

Old way to define your database is:

```
DATABASE_ENGINE = 'postgresql_psycopg2'
DATABASE_NAME = 'chimere'
DATABASE_USER = 'chimere-user'
DATABASE_PASSWORD = 'password'
DATABASE_HOST = 'localhost'
DATABASE_PORT = ''
```

The new one looks like:

```
DATABASES = {
    'default': {
    'NAME': 'ratatouille',
    'ENGINE': 'django.contrib.gis.db.backends.postgis',
    'HOST': 'localhost',
    'PORT': '5432',
    'USER': 'chimere-user',
    'PASSWORD': 'password',
    },
}
```

Be careful to adapt properly your settings.py

2.5 Run migration scripts

Migration scripts test your installation before making changes so you probably won't have any lost but by precaution before running theses scripts don't forget to backup your database. You can also make a copy of your current database into a new database and make the new installation to this new database.

The gdal binding for python is necessary to run the upgrade scripts (available in the python-gdal package in Debian).

If you run the migration scripts in a production environnement stop the old instance of Chimère before executing the migration script. Perhaps prepare the web server to point to the new installation before doing the database upgrade (cf. next paragraph).

In "settings.py" verify that "chimere.scripts" is in the INSTALLED_APPS.

After that in the chimere directory just execute the script:

```
python ./scripts/upgrade
```

2.6 Point to the new installation

Most of the job is done. You'll just have to configure your web server to serve the new version.

For instance for Apache the directive is changed from:

```
PythonPath "['/var/local/django/chimere/'] + sys.path"
```

To:

```
PythonPath "['/var/local/django/chimere-1.0.2/'] + sys.path"
```

Restart your web server and apart from web browser cache issues this should work.

2.7 Force the upgrade of visitor's web browser cache

If major changes in the javascript has be done between version, many of your users could experience problems. There are many tricks to force the refresh of their cache. One of them is to change the location of statics files. To do that edit your settings.py and change:

```
MEDIA_ROOT = ROOT_PATH + 'static/'
MEDIA_URL = '/' + EXTRA_URL + 'static/'
```

To:

```
MEDIA_ROOT = ROOT_PATH + 'static/v1.0.2/'
MEDIA_URL = '/' + EXTRA_URL + 'static/v1.0.2/'
```

Then in the static directory:

```
ln -s `pwd` v1.0.2
```

Restart the web server to apply changes.