
Chimère Documentation

Release 2.0

Étienne Loks

July 17, 2014

1	Installation	1
1.1	Prerequisites	1
1.2	Database configuration	2
1.3	Installing the sources	2
1.4	Creating a custom project template	4
1.5	Compiling languages	5
1.6	Database initialisation	5
1.7	Webserver configuration	6
1.8	Configuring the Sites framework	6
2	Upgrade	7
2.1	Getting new versions of dependencies	7
2.2	Getting the new sources	8
2.3	Migrate database	10
2.4	Update translations	11
2.5	Forcing the refresh of visitor's web browser cache	12
2.6	Configuring the Sites framework	12
3	Configuration	13
3.1	Managing areas	13
3.2	Managing users	14
3.3	Creating property models	15
4	Administration	17
4.1	Administration pages presentation	17
4.2	Managing news	21
4.3	Creating categories/subcategories	21
4.4	Editing or moderating items	21
4.5	Managing end user amendment/imported item modified locally	22
5	Import/export	25
5.1	Importing	25
5.2	Exporting	30
6	Customisation	33
6.1	Managing layers	33
6.2	Customizing the layout and the design	33

Installation

Author Étienne Loks

date 2013-03-16

Copyright CC-BY 3.0

This document presents the installation of Chimère.

1.1 Prerequisites

If you want to install the Chimère package for Debian Wheezy dependencies are managed by the package. You can go to the next section of the documentation.

- Apache version 2.x
- Python versions 2.6 or 2.7
- Django >= version 1.4
- South
- Postgres >= version 8.x
- Gettext
- Psycopg2
- Python Imaging Library
- Pyexiv2
- Beautiful Soup
- python-simplejson
- python-gdal
- Lxml
- JQuery version 1.7.1 or better
- JQuery-ui
- Universal Feed Parser

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

- [geos 3.0.x](#)
- [proj.4 4.4 to 4.6](#)
- [postgis versions 1.2.1 or 1.3.x](#)
- [gdal](#)

Optionaly (but recommended):

- [tinymce](#)
- [gpsbabel](#)
- [django-celery](#) if you want to manage large imports

The simplest way to obtain these packages is to get them from your favorite Linux distribution repositories. For instance on Debian Wheezy:

```
apt-get install apache2 python python-django python-django-south \  
  postgresql-9.1 gettext python-psycopg2 python-imaging \  
  python-pyexiv2 python-beautifulsoup python-simplejson python-gdal \  
  python-lxml libjs-jquery libjs-jquery-ui python-feedparser \  
  libgeos-3.3.3 proj-bin postgresql-9.1-postgis gdal-bin \  
  tinymce gpsbabel python-django-celery javascript-common
```

On Debian Squeeze (you need to activate backports):

```
apt-get install -t squeeze-backports python-django libjs-jquery  
  
apt-get install apache2 python python-django python-django-south \  
  postgresql-8.4 gettext python-psycopg2 python-imaging \  
  python-pyexiv2 python-beautifulsoup python-simplejson python-gdal \  
  python-lxml libjs-jquery libjs-jquery-ui python-feedparser \  
  libgeos-3.2.0 proj-bin postgresql-8.4-postgis gdal-bin \  
  tinymce gpsbabel javascript-common
```

The package *python-django-celery* doesn't exist for Debian Squeeze.

If these packages do not exist in your distribution's repository, please refer to the applications' websites.

1.2 Database configuration

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

```
su postgres  
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):

```
PG_VERSION=9.1 # 8.4 for debian Squeeze  
createdb --echo --owner chimere-user --encoding UNICODE chimere "My Chimère database"  
createlang plpgsql chimere # only necessary on Debian Squeeze  
psql -d chimere -f /usr/share/postgresql/$PG_VERSION/contrib/postgis-1.5/postgis.sql  
psql -d chimere -f /usr/share/postgresql/$PG_VERSION/contrib/postgis-1.5/spatial_ref_sys.sql
```

1.3 Installing the sources

Note: If you are considering to contribute on Chimère get the Git master.

Choose a path to install your Chimère:

```
INSTALL_PATH=/var/local/django
mkdir $INSTALL_PATH
```

1.3.1 From Debian package

If you want to install the last stable version of Chimère and your system is under Debian Wheezy it is wise to use Chimère Debian packages.

You can install Chimère this way.

```
# add Chimère repository
echo "deb http://debian.peacefrogs.net wheezy main" >> /etc/apt/sources.list
apt-get update
# install
apt-get install python-django-chimere
```

1.3.2 From an archive

The last “stable” version is available in this [directory](#). 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):

```
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
```

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

```
cd $INSTALL_PATH
tar xvjf chimere-last.tar.bz2
chown -R myusername:www-data chimere
```

1.3.3 From the Git repository

Another solution is to get it from the Git repository:

```
CHIMERE_LOCALNAME=mychimere
CHIMERE_BRANCH=v2.0 # choose v2.0 for stable ou master for bleeding edge
cd $INSTALL_PATH
git clone git://www.peacefrogs.net/git/chimere
cd chimere
git checkout origin/$CHIMERE_BRANCH
```

1.4 Creating a custom project template

A default project can be found on [Gitorious](#). Get it and start a new project with it (or get another project based on Chimère):

```
cd $INSTALL_PATH/chimere
git clone git://gitorious.org/chimere-example-project/chimere-example-project.git
django-admin startproject --template=chimere-example-project mychimere_project
rm -rf chimere-example-project
```

Your project name is used for the name of the Python package of your template. As a Python package it should follow the rule of Python variable name: it must contain at least one letter and can have a string of numbers, letters and underscores (“_”) to any length. Don’t use accentuated letters. Don’t begin the name by “_” because it has special significance in Python.

In your Chimère application directory create *local_settings.py* to fit to your configuration. A base template is provided (*local_settings.py.example*) and short descriptions of the more relevant fields are given below (at least check them). Most of these settings are initialized in *settings.py*.

```
cd $INSTALL_PATH/chimere/mychimere_project
cp local_settings.py.sample local_settings.py
vim local_settings.py
```

Fields

- **DATABASES**: parameters for the database
- **PROJECT_NAME**: name of the project
- **SECRET_KEY**: a secret key for a particular Django installation. This is used to provide cryptographic signing, and should be set to a unique, unpredictable value. **Change it!**
- **ROOT_URLCONF**: url configuration for your project something like: ‘mychimere_project.urls’
- **EMAIL_HOST**: smtp of an email server to send emails
- **TINYMCE_URL**: url to tinymce path (default is appropriate for a Debian installation with tinymce package installed)
- **JQUERY_JS_URLS**: list of jquery and jquery-ui javascript urls (default is appropriate for a Debian installation with libjs-jquery libjs-jquery-ui packages installed)
- **JQUERY_CSS_URLS**: list of jquery and jquery-ui CSS urls (default is appropriate for a Debian installation with libjs-jquery libjs-jquery-ui packages installed)
- **GPSBABEL**: path to gpsbabel (default is appropriate for a Debian installation with gpsbabel package installed)
- **TIME_ZONE**: local time zone for this installation
- **LANGUAGE_CODE**: language code for this installation

Manage media path permission:

```
cd $INSTALL_PATH/chimere/mychimere_project
chown -R user:www-data media
chmod -R g+w media
```

Create log file:


```
mkdir /var/log/django
touch /var/log/django/chimere.log
chown -R root:www-data /var/log/django/
chmod -R g+w /var/log/django/
```

Regroup static files in one path:

```
cd $INSTALL_PATH/chimere/mychimere_project
./manage.py collectstatic
```

1.5 Compiling languages

If your language is available in the directory *chimere/locale/*, you will just need to get it compiled. This can be done with the following command (here, **fr** stands for French, replace it with the appropriate language code):

```
cd $INSTALL_PATH/chimere/chimere/
django-admin compilemessages -l fr
```

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

You first need to create the default po file (of course, replace **fr** according to the language you choose to create):

```
django-admin makemessages -l fr
```

There should now be a *django.po* file in *locale/fr/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.

1.6 Database initialisation

Create the appropriate tables (still being in your Chimère project directory):

```
cd $INSTALL_PATH/chimere/mychimere_project
./manage.py syncdb
```

You will be prompted for the creation of an administrator account (administration can be found at: http://where_is_chimere/admin/). Then you have to create tables managed with Django-South:

```
./manage.py migrate
```

The database is set, congratulations!

You can load the default group permissions (it is at least a good start):

```
./manage.py loaddata ../chimere/fixtures/auth_group.json
```

If you want to populate your installation with default data (don't do this on an already populated instance!):

```
./manage.py loaddata ../chimere/fixtures/default_data.json
```

1.7 Webserver configuration

1.7.1 Apache configuration with mod_wsgi

Install *mod_wsgi* for Apache:

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

Create and edit a configuration for Chimère:

```
cp $INSTALL_PATH/chimere/apache/django.wsgi \
    $INSTALL_PATH/chimere/apache/mydjango.wsgi
vim $INSTALL_PATH/chimere/apache/mydjango.wsgi
cp $INSTALL_PATH/chimere/apache/apache-wsgi.conf \
    /etc/apache2/sites-available/chimere
vim /etc/apache2/sites-available/chimere
# create log dir
mkdir /var/log/apache2/chimere/
chown www-data /var/log/apache2/chimere/
```

Adapt the files *mydjango.wsgi* (with the correct module) and Apache *chimere* (with the correct servername and correct paths).

To activate the website, reload apache:

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

If you encounter problem with the upload of files with Unicode chars in their names, activate the appropriate locale in Apache. On a Debian server with UTF-8 as default encoding, in the file */etc/apache2/envvars* uncomment the following line:

```
. /etc/default/locale
```

1.8 Configuring the Sites framework

Sites framework allow you to serve the same content on different domains. Most of you will probably use only one domain but this unique domain has to be configured. This is done in the web administration interface in *Sites > Sites*. You only need to change *example.com* by your domain name. If you forget to do that, some functionalities such as RSS feeds will not work properly.

Author Étienne Loks

date 2013-03-16

Copyright CC-BY 3.0

This document presents the upgrade of Chimère.

Warning: Before any upgrade backup the database and all your installation files (specially if you have made changes to them).

The process for migration requires a basic knowledge of Git and Linux CLI. It is *not* an easy process. A work is currently done to easy the upgrade in later versions (>2.0) of Chimère.

If several versions have been published, you should repeat all upgrading steps. For instance to upgrade from v1.1 to v2.0 you should first upgrade to v1.2 then to v2.0. The only optional step is the integration of your customisations.

The current stable version is 2.0.

Note: If you are considering to contribute on Chimère get the Git master.

The instructions are given for Debian Squeeze and Debian Wheezy.

2.1 Getting new versions of dependencies

If you want to install the Chimère package for Debian Wheezy dependencies are managed by the package. You can go to the next section of the documentation.

2.1.1 Version 1.1 -> 1.2

```
apt-get install python-lxml libjs-jquery gpsbabel python-gdal
```

2.1.2 Version 1.2 -> 2.0

Debian Squeeze

Activate the backports: <http://backports-master.debian.org/Instructions/> Then install the new dependencies:

```
apt-get install -t squeeze-backports python-django python-django-south \  
python-simplejson libjs-jquery-ui python-pyexiv2 \  
python-feedparser javascript-common libjs-jquery
```

Debian Wheezy

```
apt-get install python-django-south python-simplejson libjs-jquery-ui \  
python-pyexiv2 python-feedparser javascript-common
```

If you are planning to do major import consider the install of **Celery**.

```
apt-get install python-django-celery python-kombu
```

2.2 Getting the new sources

To simplify further instructions, some environment variables are initialized.

```
CHIMERE_PATH=/srv/chimere  
CHIMERE_BRANCH=v1.2          # version 1.1 -> 1.2  
CHIMERE_BRANCH=v2.0          # version 1.2 -> 2.0  
CHIMERE_BRANCH=master        # version 2.0 -> master  
CHIMERE_LOCALNAME=mychimere
```

Your local name is used for the name of your local Git branch and the Python package. As a Python package it should follow the rule of Python variable name: it must be at least one letter and can have a string of numbers, letters and underscores (“_”) to any length. Don’t begin the name by “_” because it has special significance in Python.

2.2.1 From Debian package

If you want to install the last stable version of Chimère and your system is under Debian Wheezy it is wise to use Chimère Debian packages.

If you have a previous installation from sources remove all chimère libraries but **keep** your project dir.

```
rm -rf $CHIMERE_PATH/chimere
```

Then you can install Chimère.

```
# add Chimère repository  
echo "deb http://debian.peacefrogs.net wheezy main" >> /etc/apt/sources.list  
apt-get update  
# install  
apt-get install python-django-chimere
```

2.2.2 Installation from sources

First of all you have to get the new version of the source code. For the upgrade process, the source code has to be from the Git repository.

From a previous Git installation

```
cd $CHIMERE_PATH
git stash # if you have uncommitted changes
git checkout origin/$CHIMERE_BRANCH -b $CHIMERE_LOCALNAME
```

From a previous tarball installation

First remove your old installation and get the Git version:

```
cd $CHIMERE_PATH
cd ..
rm -rf $CHIMERE_PATH
git clone git://www.peacefrogs.net/git/chimere
cd chimere
git checkout origin/$CHIMERE_BRANCH -b $CHIMERE_LOCALNAME
```

2.2.3 Update basic settings

Version 1.1 -> 1.2

```
CHIMERE_APP_PATH=$CHIMERE_PATH/chimere
vim $CHIMERE_APP_PATH/settings.py
```

Add the following lines (adapted for your jquery and gpsbabel installation):

```
JQUERY_URL = SERVER_URL + 'jquery/jquery-1.4.4.min.js'
GPSBABEL = '/usr/bin/gpsbabel'
# simplify with an error of 5 meters
GPSBABEL_OPTIONS = 'simplify,crosstrack,error=0.005k'
```

Version 1.2 -> 2.0

Project template

A default project can be found on [Gitorious](#). Get it and start a new project with it (or get another project based on Chimère):

```
cd $CHIMERE_PATH
git clone git://gitorious.org/chimere-example-project/chimere-example-project.git
django-admin startproject --template=chimere-example-project mychimere_project
rm -rf chimere-example-project
```

local_settings

A *local_settings* file is now used.

```
cd $CHIMERE_APP_PATH
cp local_settings.py.sample local_settings.py
vim local_settings.py
```

Report your old settings from *settings.py* to *local_settings.py* (at least the database configuration). The setting *ROOT_URLCONF* must be set to **value_of_your_localname.urls**.

logs

Logging is now enabled by default in the file `/var/log/django/chimere.log`.

```
mkdir /var/log/django
touch /var/log/django/chimere.log
chown www-data -R /var/log/django
```

Static files

Now static files are managed with `django.contrib.staticfiles`.

```
cd $CHIMERE_APP_PATH
./manage.py collectstatic
```

Move old static files to the new static directory:

```
cp -ra $CHIMERE_PATH/chimere/static/* $CHIMERE_APP_PATH/static/
cp -ra $CHIMERE_PATH/chimere/static/icons/* $CHIMERE_APP_PATH/media/icons/
cp -ra $CHIMERE_PATH/chimere/static/upload $CHIMERE_APP_PATH/media/
rm -rf $CHIMERE_PATH/chimere/static/icons
rm -rf $CHIMERE_PATH/chimere/static/upload
```

Update permissions for media directory:

```
chown www-data -R $CHIMERE_APP_PATH/media/
```

Webserver configuration

If you are using Apache and WSGI to serve your Chimère, change your WSGI configuration file to point to the correct settings: `value_of_your_localname.settings`.

Change your webserver directive to point to the correct static directory from `your_chimere_path/chimere/static` to `your_chimere_path/your_local_name/static`.

Version 2.0 -> master

Update settings and static files.

```
cp $CHIMERE_PATH/example_project/settings.py $CHIMERE_LOCALNAME
./manage.py collectstatic
```

2.3 Migrate database

2.3.1 Version 1.1 -> 1.2

Migration scripts test your installation before making changes so you probably won't have any lost but by precaution before running these 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 environment stop the old instance of Chimère before executing the migration script.

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

After that in the Chimère directory just execute the script.

```
cd $CHIMERE_APP_PATH
python ./scripts/upgrade.py
```

2.3.2 Version 1.2 -> 2.0

Django South is now used to manage database migrations.

```
cd $CHIMERE_APP_PATH
./manage.py syncdb --noinput
./manage.py migrate chimere 0001 --fake # fake the database initialisation
./manage.py migrate chimere
```

A description field is now available for markers. If you would like to move values of an old *Property model* to this new field, a script is available.

```
cd $CHIMERE_APP_PATH
../chimere/scripts/migrate_properties.py
# follow the instructions
```

2.3.3 Version 2.0 -> master

```
cd $CHIMERE_APP_PATH
./manage.py syncdb
# les migrations ont été réinitialisées
./manage.py migrate chimere --delete-ghost-migrations --fake 0001
./manage.py migrate chimere
```

2.4 Update translations

2.4.1 Version 1.1 -> 1.2

```
cd $CHIMERE_APP_PATH
./manage.py compilemessages
```

2.4.2 Version 1.2 -> 2.0 -> master

```
cd $CHIMERE_PATH/chimere
django-admin compilemessages
```

2.5 Forcing the refresh of visitor's web browser cache

Major changes in the javascript has been done between versions, 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 `local_settings.py` and change:

```
STATIC_URL = '/static/'
```

to:

```
STATIC_URL = '/static-v2.0.0/'
```

Then change the webserver directive to point to your new path. Restart the web server to apply this changes.

2.6 Configuring the Sites framework

2.6.1 Version 1.2 -> 2.0

Sites framework allow you to serve the same content on different domains. Most of you will probably use only one domain but this unique domain has to be configured. This is done in the web administration interface in *Sites* > *Sites*. You only need to change *example.com* by your domain name. If you forget to do that, some functionalities such as RSS feeds will not work properly.

Configuration

Author Étienne Loks

date 2013-02-01

Copyright CC-BY 3.0

This document presents the first steps to configure your Chimère. It has been updated for version 2.0.0.

Your session has to be initialised with these environment variables in the Command Line Interface:

```
CHIMERE_PATH=/srv/chimere # change with your installation path
CHIMERE_LOCALNAME=mychimere # change with your local project name
CHIMERE_APP_PATH=$CHIMERE_PATH/$CHIMERE_LOCALNAME
```

Once the application installed, there are a few simple steps to follow to configure *your* Chimère.

Most of these steps are done in the web administration pages.

If you are not familiar with *Django-like* administration pages you can look at the first paragraph of [Administration](#) where it is presented.

To access these pages you have to identify with an account with *staff* and *superuser* status.

A *superuser* account is created at the initialization of the database.

3.1 Managing areas

An *Area* is the base of your map. It defines:

- a name: a human readable label for this area,
- an associated URN (*not mandatory*): the name of the area as a web ressource. In practice, if the area is not the default area the URN is used at the end of the default URL to access to this area. This is not mandatory but necessary for each area that is not the default one,
- a welcome message (*not mandatory*): this message is displayed once a day per user arriving on the map,
- an order (to sort areas),
- an availability,
- a “*default*” state. The “*default*” area is viewed by default. Only one area can be the default: activating this state disables it on the possible other area with a default state,
- default checked categories (*not mandatory*),

- if categories are displayed dynamically. If dynamically is set, the end user only views categories which have items on the map section he is currently looking at,
- categories restriction (*not mandatory*): if no restriction is set all categories are available,
- an external CSS file (*not mandatory*): an URL to an external CSS file,
- restriction on the bounding box: if set to restricted, the end user can't pan outside the defined bounding box. Due to technical reasons of OpenLayers, there is at this time no restriction on the zoom,
- a map bounding box: this is the area to display when arriving on the map. If the area is restricted it will be the bounding box that restricts the end user. Hold the *control* key, click and drag to draw the bounding box,
- available layers (*not mandatory*: OSM Mapnik is used by default): OSM Mapnik render, OSM MapQuest render, OSM Transport Map render, OSM CycleMap are available by default. You can add new custom layers (cf. *Managing layers*).

Areas are customizable directly on the web administration interface in *Chimere > Areas*.

As there is little chance that the default area should be appropriated for you, you'll have to set at least one default area.

Adding many areas can be a mean to show your map in different flavors.

3.2 Managing users

If you are not the only administrator/moderator of this Chimère installation you have to create and manage account for the other users.

You can create a new *superuser* account with the Command Line Interface (CLI):

```
./manage.py createsuperuser
```

User password can be changed with the CLI (useful if you have forgotten your password):

```
./manage.py changepassword username
```

Users are customizable directly on the web administration interface in *Auth/User*.

To create a new account, simply click on the *Add* button near *Users*. Give a name and a default password (the user can change it on his own later).

Then complete the other pieces of information.

Check the case: *Staff status* (or this user will not be able to log to the administration website).

If this account is a new technical administrator, check *Superuser status* (this user must be trustworthy!). Otherwise you'll have to give permissions to this new user. It is easier not to add permission manually but to make this user a member of a group.

Two types of default group are proposed: application administrator and moderator.

Moderator are limited to an *Area* (they only see items that are inside the bounding box). If a moderator manages many areas you'll have to select many groups.

Detail of rights for default groups:

Item (add/modify/delete on)	Technical administrator	Application administrator	Moderator
User	yes	no	no
Group	yes	no	no
Property model	yes	no	no
Import	yes	no	no
Layer	yes	no	no
News	yes	yes	no
Area	yes	yes	no
Icon	yes	yes	no
Color/Color theme	yes	yes	no
Category/Subcategory	yes	yes	no
Point Of Interest	yes	yes	yes
Route	yes	yes	yes

3.3 Creating property models

A basic installation of Chimère permits to associate a name, a category, a description, dates, multimedia files, picture files, etc. for each geographic item.

You may want to add more custom fields like phone number or opening hours. For that all you have to do is to add a new property model (*Chimere/Property model*).

The administration page asks you for:

- a name,
- an order (to sort properties),
- an availability to the end user (this can be used to set hidden properties),
- a mandatory status,
- the categories the property applies to (if no categories selected it applies to all),
- the type: text, long text, password or date.

To make this property available it is necessary to reload your web server (the property is cached).

All forms are then automatically updated with this new field.

If you don't want to allow add and modification of properties you can disable this form by setting `CHIMERE_HIDE_PROPERTYMODEL` to *True* in your *local_settings.py* file.

Administration

Author Étienne Loks

date 2012-11-28

Copyright CC-BY 3.0

This document presents the administration of Chimère. It has been updated for version 2.0.0.

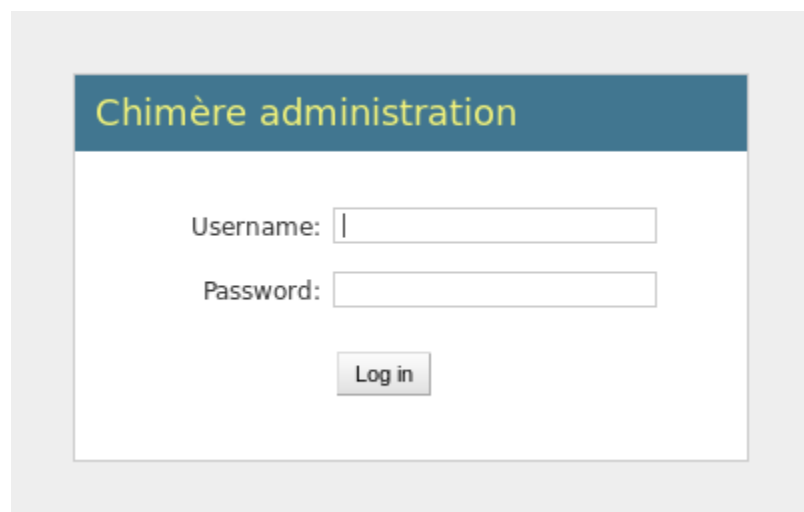
4.1 Administration pages presentation

Administration pages are accessible at: http://where_is_your_chimere/admin/

Don't forget the ending slash in the URL.

4.1.1 Identification

First of all, you'll have to identify yourself with the login and password provided.

The image shows a screenshot of the Chimère administration login page. At the top, there is a dark blue header with the text "Chimère administration" in a light green font. Below the header, there is a white form area. The form contains two input fields: "Username:" followed by a text input box, and "Password:" followed by a password input box. Below these fields is a "Log in" button with a grey background and white text.

4.1.2 Main page

Once authenticated you'll have access to the main admin page.

It looks like this:

The screenshot displays the Chimère administration interface. At the top, a dark blue header contains the text 'Chimère administration' on the left and 'Welcome, **admin**. Documentation / Change password / Log out' on the right. A yellow circle with the number '1' is positioned above the 'Log out' link. Below the header, the main content area is titled 'Site administration'. It features a table with various categories, each with 'Add' and 'Change' links. A yellow circle with the number '3' is placed above the 'Chimere' category. A yellow circle with the number '4' is placed above the 'Categories' category. To the right of the main table is a 'Recent Actions' sidebar, which is titled 'My Actions' and contains a list of actions performed by the user, such as 'Application administrator', 'Vern-sur-Seiche moderation', and 'Pays de Rennes moderation'. A yellow circle with the number '2' is placed above the 'Recent Actions' title.

1. links to this **Documentation**, to the **Change password** form and to **Log out**,
2. the list of recent actions made with this account,
3. an application title, most of your action will be in the **Chimere** application,
4. an item inside the application. From these pages you can **Add** a new item or consult/**Change** items. The **Add** link leads to the new **Item form**. The **Change** link leads to the **Item list**. The **Item list** is also available by clicking on the item label.

4.1.3 Item list

Chimère administration Welcome, **admin**. Documentation / Change password / Log out

Home > Chimere > Point of interests **1**

Select Point of interest to change **2** Add Point of interest +

3 Search 6 results (625 total)

Action:
Go **7** of 6 selected

<input type="checkbox"/>	5 Name 2 ▲	1 Status ▲
<input type="checkbox"/>	Allen Hall classic	Imported
<input type="checkbox"/>	Allen Hall Pas Modif	Imported
<input type="checkbox"/>	Austin Hot Springs, OR 6	Imported
<input type="checkbox"/>	Bagby Hot Springs, OR	Imported
<input type="checkbox"/>	Bechler River Hot Springs, WY	Imported
<input type="checkbox"/>	Breitenbush Hot Springs, OR	Imported

6 Point of interests

Filter **4**

By Status

- All
- Submitted
- Available
- Modified
- Disabled
- Imported
- Excluded

By categories

- All
- Default category /
- Default subcategory
- Patrimoine / Test
- d'itinéraire
- Test / Test

1. path in the admin site. This is a convenient shortcut to come back to the main page,
2. link to create a new item from the item list,
3. search items by words (not available for all item types),
4. this filter box permits to filter current entries with some criteria (not available for all type of items),
5. the header of the table is clickable. Clicking on an header sorts the items by this header (ascending or descending). Multiple header sort is possible (the number on the right of the header explains the sorting order),
6. each item can be checked (for applying an action) or selected (by clicking on the first column) to see the detail and edit or delete it.

4.1.4 Item form

Chimère administration
Welcome, **admin**. [Documentation](#) / [Change password](#) / [Log out](#)

[Home](#) > [Chimere](#) > [Categorys](#) > [Default category](#)

Change Category

History

Name:

Available 1

Order:

Description:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc blandit porta eros, quis varius orci luctus nec. Suspendisse tempor sagittis tortor. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vivamus imperdiet consequat dolor. Etiam risus orci, auctor nec hendrerit ut, congue at ante. Aliquam et accumsan neque. Aliquam ac massa felis, ac porttitor nunc. Curabitur blandit odio id enim sodales blandit. Curabitur eleifend commodo lorem a feugiat. Nam

Sub-categories 2

Name	Available	Available for submission	Icon	Color theme	Item type	Delete?
Default category / Default subcategory <input type="text" value="Default subcategory"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Default icon	-----	Marker	<input type="checkbox"/>
<input type="text"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	-----	-----	-----	

[+ Add another Sub-Category](#)

✖ Delete
Save and add another
Save and continue editing
Save

1. fields for the selected item (or blank if it is a new item) are displayed in this form. A few fields are read-only and another few are hidden. Mandatory fields have their label in bold. Changes on these fields are only effective once the form is submitted.
2. for some items there are associated sub-items. These associated items can be modified in this form. If there are many sub-items associated for the current item, they can be ordered by drag and drop.
3. the form has to be validated by one of these action buttons. They are self-explanatory.

4.1.5 Status

Status is a property attached to each geographic item in Chimère. To administrate efficiently Chimère you need to understand the mean of each status.

- **Submitted:** Status of a new item freshly proposed by an end user. This item is not visible on the map.
- **Available:** Status of an item visible on the map.
- **Disabled:** Status of a discarded item.

- **Modified:** Status of an amendment proposed by an end-user.
- **Imported:** Status of a newly imported item. Import and export operations need that all items with *imported* status are treated (validated, disabled or deleted).

4.2 Managing news

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

4.3 Creating categories/subcategories

Before adding categories you have to set some icons. These icons appear 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.

The website <http://mapicons.nicolasmollet.com/> allow to easily generate map 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. These fields are mainly quite self-explanatory. 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 that the subcategory can contain: POI, route or both.

4.4 Editing or moderating items

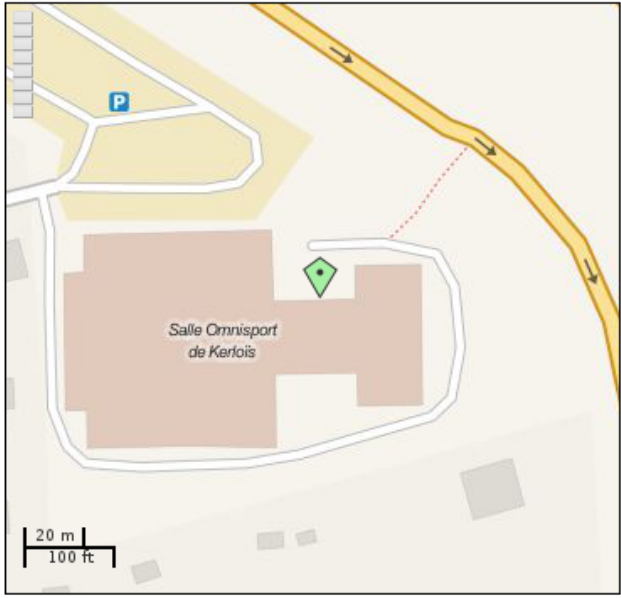
The moderation step is quite simple. It works the same with POIs and routes. The moderator can access to all POIs (or routes) by clicking on them on the list.

A search field is available to search by name but the more interesting is to filter POIs (or route) by state and by subcategory.

There are some actions available in the action list:

- **Delete:** to delete selected items. A confirmation step is displayed,
- **Validate:** to set the status *Available* to selected items,
- **Disable:** to set the status *Disabled* to selected items. This is useful to keep items you don't want to be exposed on the map,
- **Managed modified items:** to manage the amendment made by end users on the main site (cf. *Managing end user amendment/imported item modified locally*). Modified item has to be treated one by one,
- **Export to...:** to export selected item to the selected format.

To modify an item, classically you have to click on its name. Then you access to a form to freely modify the item.

Name: <input type="text" value="Aikido Shobukan Gouesnou"/>	Localisation: 	
Status: Available ▾		
Categories: Select... ▾ Default category / Imports kml X		
Description: <input type="text" value="Click here to open the article page in your browser"/>		
Start date: <input type="text"/> Not mandatory. Set it for dated item such as event. Format YYYY-MM-DD		
End date: <input type="text"/> Not mandatory. Set it only if you have a multi-day event. Format YYYY-MM-DD		
Submitter (Afficher)		
Import (Afficher)		
Associated items (Afficher)		
Multimedia files		
Name	Url	Multime

In this form there are all data available to the end user form plus some extra fields.

- The *Import fields* only make sense with data imported from an external source or for data to be exported to OSM (cf. the *import section* of this documentation),
- *Associated items fields* are read-only fields that list items associated to the current one (for example the reference item of an amendment or an associated file of a route).

Associated multimedia items are at the bottom of the form. You can freely add, change items and change their order with drag and drop.

Don't forget to validate your change with one of the **Save** buttons at the bottom of the form (it is easily forgotten when you change multimedia items).

If an item is not relevant the **Delete** button enables to remove it.

4.5 Managing end user amendment/imported item modified locally

Amendment can be proposed on the main site by end-users. In Chimère an amendment is a new item with the status **Modified** and with a link to the reference item modified.



You can also have imported items which have been modified both locally and on the external source. The new version from the external source has the status **Imported** and have a link to the reference item.

Note: If you are logged as an administrator and make changes on the map with the end user form they will be directly validated.

A special form has been developed to facilitate the processing of these modified items.

You can access to this special form with the action *Managed modified items*.

⚠ Be careful: after validation, the modified item will be deleted. There is no roll-back.

	Reference 1	Modified item 2	Accept modification 3
Name	La Petite Chaumière	La Petite Chaumière	<input checked="" type="checkbox"/>
Categories	Default category / Imports osm	Default category / Imports osm	<input type="checkbox"/>
Emplacement			<input checked="" type="checkbox"/>
Description			<input type="checkbox"/>

[Back to list](#)

This form is a table with three columns:

1. The first column displays the information for the reference item,
2. The second column displays the information proposed by the submitter,
3. The third column is a list of checkboxes. For each row checked, after the validation, the value of the modified item will replace the value of the reference item.

Note: To reject all modifications validate the form with no checkbox checked.

Import/export

Author Étienne Loks

date 2013-02-01

Copyright CC-BY 3.0

This document presents the import/export functionalities of Chimère. It has been updated for version 2.0.0.

5.1 Importing

In Chimère the import mechanism is based on **Import object**. These objects are stored in database to keep trace of imports and to facilitate the re-importation from the same source. In fact if possible the update of data from a same type of source is managed.

Note: The ability to do such updates depends on the existence of a unique id for each object on your source.

To add an **Import object** you need to go to *Chimere > Imports* then **Add**.

After that you'll have to select your source type and then the form depends on this source type.

5.1.1 Common fields

- **Name by default:** if no name can be identified to the newly imported object this is the name that will be used. If this field is empty the name of the associated category will be use.
- **SRID:** Chimère will try to identify automatically the correct coordinate system from the given source. But sometimes the information is not present or cannot be guessed (for instance a Shapefile that uses non standard proj file). In this case Chimère will use WGS84 by default (the classic latitude/longitude) but it is not always correct. If you experience problems with items localisation you should put here the **SRID** associated to the coordinate system of your source.
- **Overload existing data:** by default when data is updated on the source and has been changed in your Chimère instance a new item is created and has to be reconciled with the *amendment form*. If you don't want to use this form and then overwrite with the data from the external source, check this option.
- **Get description from source:** If this case is checked, the importer will try to get the description from the source file. This option is only available for certain file formats.
- **Default description:** A default description to be added to new items. This field is only available when **Get description from source** is not checked.

- **Origin:** if not null this field will be associated to each item imported in order to easily identify where the item came from. For OSM import the source is automatically added.
- **License:** if not null this field will be associated to each item imported in order to easily identify the license associated to the item. For OSM import the license is automatically added.
- **Associated subcategories (mandatory):** The selected subcategories will be associated to newly imported items.

5.1.2 KML import

Importer type:	<input type="text" value="KML"/>
Filter:	<input type="text"/> <small>You can put a Folder name of the KML file to filter on it.</small>
Web address:	<input type="text"/>
Source file:	<input type="text"/> <input type="button" value="Parcourir..."/>
Name by default:	<input type="text"/>
<input type="checkbox"/> Zipped file	
<input type="checkbox"/> Overwrite existing data	
Origin:	<input type="text"/>
License:	<input type="text"/>
Associated subcategories:	<input type="text" value="Select..."/>

- **Web address/source file (mandatory):** your KML could be distant or a local file. You'll have to fill one of the two fields.
- **Filter:** if you want to import only a specific *Folder* of your KML file put his name on this field.
- **Zipped file:** if your source is a KMLZ file (a zipped KML), check this case.

5.1.3 Shapefile import

Importer type:	<input type="text" value="Shapefile"/>
Web address:	<input type="text"/>
Source file:	<input type="text"/> <input type="button" value="Parcourir..."/>
Name by default:	<input type="text"/>
SRID:	<input type="text"/>
<input type="checkbox"/> Zipped file	
<input type="checkbox"/> Overwrite existing data	
Origin:	<input type="text"/>
License:	<input type="text"/>
Associated subcategories:	<input type="text" value="Select..."/>

- **Web address/source file (mandatory):** your Shapefile could be distant or a local file. You'll have to fill one of the two fields.
- **Zipped file:** only zipped Shapefiles are accepted so this checkbox has to be checked.

5.1.4 GeoRSS import

Simple GeoRSS and W3C GeoRSS are managed.

Importer type:	<input type="text" value="GeoRSS"/>
Web address:	<input type="text"/>
Name by default:	<input type="text"/>
SRID:	<input type="text"/>
<input type="checkbox"/> Overwrite existing data	
Origin:	<input type="text"/>
License:	<input type="text"/>
Associated subcategories:	<input type="text" value="Select..."/>

- **Web address (mandatory):** only distant GeoRSS are managed.

5.1.5 CSV import

The format of the CSV file (number and order of columns) managed by Chimère varies depending on the properties you have added on your Chimère instance. So we recommend you to first do an export of some items in CSV with Chimère. The CSV format of the exported file will meet Chimère requirements.

By the way because of the geometry of the item this format is not very convenient to add new content but could be handy to update informations.

Warning: If you mean to update existing data by this import, unless you know how to edit WKT do *not* modify the geometry column.

Importer type:	<input type="text" value="CSV"/>
Web address:	<input type="text"/>
Source file:	<input type="text"/> <input style="border: none; background-color: #ccc; padding: 2px 5px;" type="button" value="Parcourir..."/>
Name by default:	<input type="text"/>
SRID:	<input type="text"/>
<input type="checkbox"/> Overwrite existing data	
Origin:	<input type="text"/>
License:	<input type="text"/>
Associated subcategories:	<input type="text" value="Select..."/>

- **Web address/source file (mandatory):** your CSV file could be distant or a local file. You'll have to fill one of the two fields.

5.1.6 OpenStreetMap import

Importer type: OSM

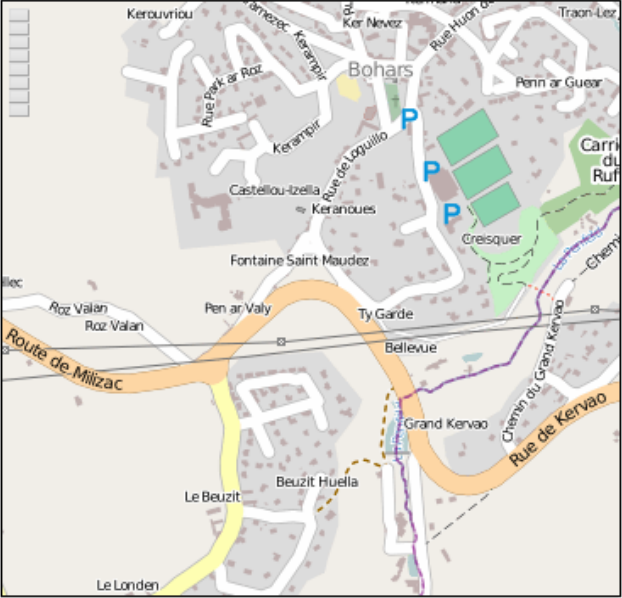
Web address:

Name by default:

Overwrite existing data

Associated subcategories: Select...

Filtr: Hold CTRL, click and drag to select area on the map



Type: Node Way

Enter an OSM "tag=value" string such as "amenity=pub". A list of common tag is available [here](#).

Tag:

⊘ If you change the above form don't forget to refresh before submit!

To import from OSM Chimère use the XAPI API of OSM.

- **Web address (mandatory):** XAPI url used to import data. This field should be filled with a default address. By default the MapQuest server is used as it seems to be the more robust. If you experience problems with OSM import, check the availability of the XAPI server used and eventually change it.
- **Filter area (mandatory):** draw the bounding box you want to use for your OSM import.
- **Filter type (mandatory):** choose if you want to import way or nodes.
- **Filter tag (mandatory):** choose the OSM key/value couple used to filter OSM data. A link to the [OSM Map features page](#) is provided to help you find appropriate values.
- **Refresh button:** this button convert your choices to appropriate XAPI args. You have to hit this button before validating the form.

5.1.7 Importing

Once your new import item created, select it in the import object list, choose the **Import** action and validate.

The import should be processing normally. If not, an explicit error message should be printed in the state column of your import.

You can also launch imports with the CLI (ideal for crontab jobs). In the project directory you only need to launch the command:

```
./manage.py chimere_import <import_id>
```

- *import_id* is the import ID

If you launch the command without *import_id* the list of imports available is presented and you can choose one.

5.1.8 Managing imported data

All new imported items have the state **Imported**. To make them available on the map you'll have to validate them. If you don't want some items to be visible on the map, instead of deleting them it is recommended to set them to the state **Disabled**. So on the next update from the source, rather than appear as new items they remain disabled.

Warning: Be careful with duplicates between your existing data and imported data. This is particularly important if you want to export your data to OSM.

5.2 Exporting

5.2.1 Export to CSV/KML/Shapefile

Directly from the *geographic items list* you can export to the chosen format. All you have to do is to select the desired items, choose the appropriate action in the action list and validate.

You can also launch exports with the CLI (ideal for crontab jobs). In the project directory you only need to launch the command:

```
./manage.py chimere_export <subcategory_id> <CSV|KML|SHP> \  
                           <marker|route> <filename>
```

- *subcategory_id* is the ID of the chosen subcategory
- *CSV|KML|SHP* is the chosen format
- *marker|route* is to get marker or route
- *filename* is the output filename

If you launch the command without arguments you will be prompted for the choice to make for your export.

5.2.2 Export to OSM

Warning: If you are not sure of what you are doing with OSM export: don't do it! It is really important to not mess with others' data.

Note: Only export of OSM nodes are managed.

OSM export is not that easily managed. First (if not yet done) you'll have to define an import (*see above* for details). This will enable to determine:

- the area concerned by your export.
- the key/value tag to append to your new/updated items.
- the subcategories concerned by your export. If you think that some items in these subcategories should not be in OSM database (because there are not relevant or because of license issues) beforehand mark them as **Not for OSM** in the *import fields* of the *geographic items forms*.

The OSM export in Chimère is designed to be the more preservative possible in regards to OSM database. That's why before any export an import is done. If the new import has updated data, treat them before doing an export (cf. *manage imported data*).

To launch an export select the appropriate *Import* object in the imports list. Then select the **Export to OSM** action and validate. Then you'll be asked for your OSM username and password and the API you want to use. If you regularly use Chimère to do export, it is recommended to create an OSM specific account for that. The test API is available to make export test. If you want to use the test API you'll have to create a specific account on the test platform.

Warning: The data on the test platform are not synced with the main platform. You won't have the same data than the ones you got with XAPI.

Once all this field filled, you can (finally!) launch the export.

When exporting tags are automatically added/updated:

- *name*: get from the item name in Chimère.
- *source*: to identify Chimère as a source.

Customisation

Author Étienne Loks

date 2012-11-28

Copyright CC-BY 3.0

This document presents the customisation of Chimère. It has been updated for version 2.0.0.

6.1 Managing layers

There are some different layers available by default in Chimère (OSM Mapnik, OSM Mapquest, OSM Transport map, OSM Cyclemap). You can add some extra layer using the web administration pages of Chimère. The new layer is defined with the appropriate [Openlayers JS](#) code. This JS code must be a compatible Openlayers Layer instance with no ending semi-colon. For instance defining a Bing layer can be done with this kind of code:

```
new OpenLayers.Layer.Bing({
    name: "Aerial",
    key: "my-bing-API-key",
    type: "Aerial"})
```

Refer to the [Openlayers documentation API](#) for more details.

6.2 Customizing the layout and the design

If you only want to customize the CSS, the easiest way to do it is to add a link to an extra CSS to your *Areas* cf. [Managing areas](#).

If you want to do larger changes in the layout and the style the (well named) `example_project` can be customized to fit your needs. Each template file present in the `chimere/templates` directory can be copied in your `myproject/templates` directory and then modified. You only need to copy files that you want to modify. These files are in Django template language mainly made of pure HTML with some logic. Refer to the [Django template documentation](#) for more details.