Documentation de Chimère

Version 2.0

Étienne Loks

17 July 2014

Table des matières

1 Installation **1**
- 1.1 Pré-requis . . . 1
- 1.2 Configuration de la base de données . . . 2
- 1.3 Installer les sources . . . 2
- 1.4 Créez un patron pour votre projet . . . 3
- 1.5 Compilation des langages . . . 4
- 1.6 Initialisation de la base de données . . . 5
- 1.7 Configuration du serveur web . . . 5
- 1.8 Configurer le framework Sites . . . 6

2 Mise à jour **7**
- 2.1 Obtenir des nouvelles versions des dépendances . . . 7
- 2.2 Obtenir les nouvelles sources . . . 8
- 2.3 Migration de la base de données . . . 11
- 2.4 Mise à jour des traductions . . . 12
- 2.5 Forcer le rafraîchissement du cache du navigateur des utilisateurs . . . 12
- 2.6 Configurer le framework Sites . . . 12

3 Configuration **13**
- 3.1 Gérer les zones . . . 13
- 3.2 Gestion des utilisateurs . . . 14
- 3.3 Créer des modèles de propriété . . . 15

4 Administration **17**
- 4.1 Présentation des pages d'administration . . . 17
- 4.2 Gestion des nouvelles . . . 21
- 4.3 Création de catégories/sous-catégories . . . 21
- 4.4 Édition/modération des éléments . . . 21
- 4.5 Gérer les modifications des utilisateurs/les éléments importés ayant des modifications locales . . . 22

5 Import/export **25**
- 5.1 Import . . . 25
- 5.2 Export . . . 30

6 Personnalisation **33**
- 6.1 Gestion des calques . . . 33
- 6.2 Personnaliser l'agencement et le design . . . 33

CHAPITRE 1

Installation

Auteur Étienne Loks

date 2013-03-16

Ce document présente l'installation de Chimère.

1.1 Pré-requis

Si vous souhaitez installer le paquet Debian prévu pour Wheezy, les dépendances sont gérées par le paquet. Vous pouvez passer à la section suivante de la documentation.

- Apache version 2.x
- Python versions 2.6 ou 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 fait partie de Django depuis la version 1.0 mais nécessite quelques dépendances supplémentaires :

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

Optionnel (mais recommandé) :

- tinymce
- gpsbabel
- django-celery pour une meilleure gestion des imports importants

La manière la plus simple de satisfaire à ces pré-prequis est de les installer par le biais des dépôts de votre distribution Linux préférée. Par exemple pour 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
```

Pour Debian Squeeze (il est nécessaire d'activer les 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
```

Le paquet *python-django-celery* n'existe pas pour Debian Squeeze.

Si ces paquets n'ont pas d'équivalents dans les dépôts de votre distribution Linux, référez-vous aux sites web de ces applications.

1.2 Configuration de la base de données

Maintenant que postgres et postgis sont installés, vous avez besoin de créer un nouvel utilisateur pour Chimère :

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

Ensuite, vous avez besoin de créer la base de données et d'initialiser les types géographiques (adaptez les chemins par rapport à vos besoins) :

```
PG_VERSION=9.1 # 8.4 pour debian Squeeze
createdb --echo --owner chimere-user --encoding UNICODE chimere "Ma base de données Chimère"
createlang plpgsql chimere # seulement nécessaire sous 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 Installer les sources

Choisissez un chemin où installer Chimère

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

1.3.1 Depuis les paquets Debian

Si vous souhaitez disposer de la dernière version stable de Debian et que vous êtes sous environnement Wheezy, il est conseillé d'utiliser les paquets prévus à cet effet.

Vous pouvez installer Chimère ainsi.

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

1.3.2 Depuis une archive

La dernière version « stable » est disponible dans ce répertoire. Prenez garde à prendre la dernière version de la branche souhaitée (par exemple la dernière version de la branche 1.0 est la 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
```

Téléchargez, décompressez et déplacez les fichiers dans un répertoire lisible par l'utilisateur de votre serveur web (www-data pour Debian).

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

1.3.3 Depuis le dépôt Git

Une autre solution est d'obtenir les sources depuis le dépôt Git :

```
CHIMERE_LOCALNAME=mychimere
CHIMERE_BRANCH=v2.0 # choisissez v2.0 ou master
cd $INSTALL_PATH
git clone git://www.peacefrogs.net/git/chimere
cd chimere
git checkout origin/$CHIMERE_BRANCH -b $CHIMERE_LOCALNAME
```

1.4 Créez un patron pour votre projet

Un exemple de projet peut être trouvé sur Gitorious. Clonez-le et modifiez-le (ou utilisez un autre projet basé sur 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
```

Le nom de votre projet est utilisé pour le nom de la bibliothèque Python correspondant à votre projet. En tant que bibliothèque Python, ce nom doit suivre les règles de nommage des noms de variable Python : il doit comporter au moins une lettre et peut comporter autant de nombres et de lettres que souhaité, le caractère tiret bas (« _ ») est accepté. N'utilisez pas de caractères accentués. Ne commencez pas par « _ » car cela a une signification particulière en Python.

Dans le répertoire de votre application Chimère créez un fichier *local_settings.py* qui correspond à votre configuration. Un fichier de base est fourni (*local_settings.py.example*) et des descriptions courtes des variables les plus pertinentes sont données sous celui-ci (survolez-les au minimum). La plupart de ces paramétrages sont initialisés dans le fichier *settings.py*.

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

Champs

- DATABASES : paramètres relatifs à la base de données
- PROJECT_NAME : nom du projet
- SECRET_KEY : une clé secrète pour l'installation de votre application Django. Cette clé est utilisée pour les signatures cryptographiques de l'application et doit être initialisée à une valeur unique et non devinable. **Modifiez-là !**
- ROOT_URLCONF : module Python de configuration des urls pour votre projet. Cela devrait être quelque chose comme : 'mychimere_project.urls'
- EMAIL_HOST : SMTP du serveur de courriel pour envoyer des courriels
- TINYMCE_URL : url du chemin vers tinymce (le chemin par défaut est adapté pour une installation sous Debian avec le paquet tinymce installé)
- JQUERY_JS_URLS : liste des adresses des fichiers javascript jquery et jquery-ui (les valeurs par défaut sont appropriées pour une installation sous Debian avec les paquets libjs-jquery et libjs-jquery-ui installés)
- JQUERY_CSS_URLS : liste des adresses des fichiers CSS jquery et jquery-ui (les valeurs par défaut sont appropriées pour une installation sous Debian avec les paquets libjs-jquery et libjs-jquery-ui installés)
- GPSBABEL : chemin de gpsbabel (la valeur par défaut est appropriée pour une installation sous Debian avec le paquet gpsbabel installé)
- TIME_ZONE : fuseau horaire local de cette installation
- LANGUAGE_CODE : code de langage pour cette installation

Gérez les permissions du dossier de média :

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

Créez le fichier de log :

```
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/
```

Regroupez les fichiers static dans un seul répertoire :

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

1.5 Compilation des langages

Si votre langage est disponible dans le dossier *chimere/locale/*, il est juste nécessaire de le compiler. Pour faire cela, il faut lancer la commande suivante (ici, **fr** est pour le français, remplacez cela avec le code de langage approprié) :

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

Si votre langage n'est pas disponible, n'hésitez pas à créer le fichier **po** par défaut et à le proposer (les contributions sont bienvenues). La procédure est explicité ci-dessous.

Il est d'abord nécessaire de créer le fichier po par défaut (bien sûr remplacez **fr** par le code du langage que vous souhaitez créer) :

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

Il doit y avoir maintenant un fichier *django.po* dans le répertoire *locale/fr/LC_MESSAGES*. Ensuite il faut le compléter avec votre traduction.

Une fois le votre fichier de traduction complété, il suffit de le compiler de la même manière que vous l'auriez fait si ce fichier était initialement disponible.

1.6 Initialisation de la base de données

Créez les tables de la base de données (toujours dans le répertoire de votre projet) :

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

Vous aurez à rentrer les informations pour la création du compte administrateur (les pages d'administration se trouvent à l'adresse : http://where_is_chimere/admin/). Ensuite pour créer les tables de la base de données gérées par Django-South :

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

La base de données est en place, félicitations !

Vous pouvez alors charger les permissions poar défaut pour les groupes (c'est au minimum un bon départ) :

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

Si vous voulez remplir votre installation avec des données par défaut (ne le faites pas sur une instance de Chimère contenant déjà des données !) :

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

1.7 Configuration du serveur web

1.7.1 Configuration d'Apache avec mod_wsgi

Installez *mod_wsgi* pour Apache :

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

Créez et éditez la configuration de Chimère en fonction de votre installation

```
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
# créer le répertoire des logs
mkdir /var/log/apache2/chimere/
chown www-data /var/log/apache2/chimere/
```

Adaptez les fichiers *mydjango.wsgi* (avec le nom correct pour le module) et le fichier *chimere* de Apache (avec le nom de serveur correct et les chemins corrects).

Pour activer le site web, rechargez Apache :

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

Si vous avez des problèmes de dépôt de fichier avec des caractères Unicode dans leurs noms, activez la locale appropriée dans Apache. Sur un serveur Debian avec UTF-8 comme codage par défaut, dans le fichier */etc/apache2/envvars* décommentez la ligne suivante :

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

1.8 Configurer le framework Sites

Le framework *Sites* vous permet de servir le contenu pour différents domaines Internet. La plupart des installations serviront le contenu pour un seul domaine mais ce domaine unique doit être configuré.

Pour cela allez dans les pages web d'administration *Sites* > *Sites*. Vous avez juste à changer *example.com* par votre nom de domaine. Si vous oubliez de faire cela, quelques fonctionnalités comme les flux RSS ne fonctionneront pas correctement.

CHAPITRE 2

Mise à jour

Auteur Étienne Loks
date 2013-03-16

Ce document présente la mise à jour de Chimère.

Warning : Avant toute mise à jour faites une sauvegarde de la base de données et de tous vos fichiers d'installation (en particulier si vous avez fait des changements sur ceux-ci).

La procédure de migration nécessite une connaissance de base de Git et de la ligne de commande Linux. Ce n'est *pas* une procédure facile. Un travail est en cours pour faciliter les mises à jour des futures versions de Chimère (>2.0).

Si plusieurs versions de Chimère ont été publiées depuis votre installation, vous devez répéter toutes les étapes de mise à jour. Par exemple pour mettre à jour depuis la version 1.1 vers la version 2.0, vous devez d'abord mettre à jour vers la version 1.2 puis vers la version 2.0. La seule étape optionnelle est l'intégration de vos personnalisations.

La version stable actuelle est la version 2.0.

Note : Si vous souhaitez améliorer Chimère prenez la branche *master* sur Git.

Les instructions sont données pour Debian Squeeze et Debian Wheezy.

2.1 Obtenir des nouvelles versions des dépendances

Si vous souhaitez installer le paquet Debian prévu pour Wheezy, les dépendances sont gérées par le paquet. Vous pouvez passer à la section suivante de la 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

Activez les backports (http://backports-master.debian.org/Instructions/). Puis installez les nouvelles dépendances

```
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
```

Si vous comptez réaliser des imports importants envisagez l'installation de Celery.

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

2.2 Obtenir les nouvelles sources

Pour simplifier les instructions suivantes, quelques variables d'environnement sont initialisées.

```
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
```

Le nom de votre projet (*CHIMERE_LOCALNAME*) est utilisé pour le nom de la bibliothèque Python correspondant à votre projet ainsi que votre propre branche Git. En tant que bibliothèque Python, ce nom doit suivre les règles de nommage des noms de variable Python : il doit comporter au moins une lettre et peut comporter autant de nombres et de lettres que souhaité, le caractère tiret bas (« _ ») est accepté. N'utilisez pas de caractères accentués. Ne commencez pas par « _ » car cela a une signification particulière en Python.

2.2.1 Depuis les paquets Debian

Si vous souhaitez disposer de la dernière version stable de Debian et que vous êtes sous environnement Wheezy, il est conseillé d'utiliser les paquets prévus à cet effet.

Si vous avez une installation précédente depuis les sources effacez les répertoires contenant la bibliothèque de base **conservez** bien les répertoires contenant votre projet.

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

Ensuite vous pouvez installer Chimère.

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

2.2.2 Installation depuis les sources

Tout d'abord vous avez besoin de la nouvelle version du code source. Pour la procédure d'installation, le code source doit être celui du dépôt Git.

Pour une précédente installation Git

```
cd $CHIMERE_PATH
git stash # si vous avez des changements pas encore « commités »
git checkout origin/$CHIMERE_BRANCH -b $CHIMERE_LOCALNAME
```

Pour une précédente installation depuis une archive

Supprimez d'abord votre ancienne installation et obtenez la version Git :

```
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 Mettre à jour les paramètres de base

Version 1.1 -> 1.2

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

Ajoutez les lignes suivantes (adaptez en fonction de vos installations jquery et gpsbabel) :

```
JQUERY_URL = SERVER_URL + 'jquery/jquery-1.4.4.min.js'
GPSBABEL = '/usr/bin/gpsbabel'
# simplification des trajets avec une tolérance de 5 mètres
GPSBABEL_OPTIONS = 'simplify,crosstrack,error=0.005k'
```

Version 1.2 -> 2.0

Patron de projet

Un exemple de projet peut être trouvé sur Gitorious. Clonez-le et modifiez-le (ou utilisez un autre projet basé sur 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

Un fichier *local_settings* est maintenant utilisé.

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

Reportez vos anciens paramètres de *settings.py* vers *local_settings.py* (au minimum la configuration de votre base de données). Le paramètre *ROOT_URLCONF* doit être mis à la valeur « **nom_de_votre_projet.urls** ».

logs

Par défaut, des fichiers de *log* sont maintenant écrit dans le fichier : « */var/log/django/chimere.log* ».

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

Fichiers statiques

Les fichiers statiques sont maintenant gérés avec « **django.contrib.staticfiles** ».

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

Déplacez vos anciens fichiers statiques vers le nouveau répertoire :

```
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
```

Mettre à jour les permissions des répertoires media :

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

Configuration du serveur Web

Si vous utilisez Apache et WSGI pour mettre à disposition votre Chimère, changez la configuration pour pointer vers le chemin correct de configuration : « **nom_de_votre_projet.settings** ».

Changez la directive de votre serveur web pour qu'elle pointe vers le bon répertoire statique de « **votre_chemin_vers_chimere/chimere/static** » en « **votre_chemin_vers_chimere/nom_de_votre_projet/static** ».

Version 2.0 -> master

Mettez à jour les paramètres et les fichiers statiques.

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

2.3 Migration de la base de données

2.3.1 Version 1.1 -> 1.2

Les scripts de migration testent votre installation avant de faire des changements. Vous n'aurez donc probablement pas de perte mais par précaution avant de les lancer n'oubliez pas de faire une sauvegarde de votre base de données. Vous pouvez aussi faire une copie de votre base de données actuelle dans une nouvelle base et faire la mise à jour sur cette nouvelle base de données.

La bibliothèque GDAL pour Python est nécessaire pour faire fonctionner ces scripts (disponible avec le paquet *python-gdal* dans Debian).

Si vous souhaitez lancer le script de migration dans un environnement de production, stoppez l'instance de Chimère avant d'exécuter le script de migration.

Dans le fichier *settings.py* vérifiez que **chimere.scripts** fait partie des *INSTALLED_APPS*.

Après cela, dans le répertoire d'installation de Chimère, exécutez simplement le script.

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

2.3.2 Version 1.2 -> 2.0

Django South est maintenant utilisé pour les migrations de base de données.

```
cd $CHIMERE_APP_PATH
./manage.py syncdb --noinput
./manage.py migrate chimere 0001 --fake # simule l'initialisation de la base
                                        # de données
./manage.py migrate chimere
```

Un champ descriptif est maintenant disponible pour les points d'intérêts. Si vous souhaitez migrer un ancien *modèle de propriété* vers ce nouveau champ, un script est disponible.

```
cd $CHIMERE_APP_PATH
../chimere/scripts/migrate_properties.py
# suivez les instructions
```

2.3.3 Version 2.0 -> master

```
cd $CHIMERE_APP_PATH
./manage.py syncdb
# migrations have been reinitialized
./manage.py migrate chimere --delete-ghost-migrations --fake 0001
./manage.py migrate chimere
```

2.4 Mise à jour des traductions

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 Forcer le rafraîchissement du cache du navigateur des utilisateurs

Des changements importants au niveau des styles et du javascript sont faits entre les différentes versions. Cela peut provoquer des dysfonctionnements importants chez des utilisateurs dont le navigateur web a conservé les anciennes versions de certains fichiers en cache. Il y a plusieurs moyens de forcer le rafraîchissement de leur cache. Un de ceux-ci est de changer le chemin vers les fichiers statiques. Pour faire cela, éditez votre fichier *local_settings.py* et changez :

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

en :

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

Changez la directive concernant les fichiers statiques sur le fichier de configuration de votre serveur web. Redémarrez alors le serveur web pour appliquer les changements.

2.6 Configurer le framework Sites

2.6.1 Version 1.2 -> 2.0

Le framework *Sites* vous permet de servir le contenu pour différents domaines Internet. La plupart des installations serviront le contenu pour un seul domaine mais ce domaine unique doit être configuré.

Pour cela allez dans les pages web d'administration *Sites* > *Sites*. Vous avez juste à changer *example.com* par votre nom de domaine. Si vous oubliez de faire cela, quelques fonctionnalités comme les flux RSS ne fonctionneront pas correctement.

CHAPITRE 3

Configuration

Auteur Étienne Loks

date 2013-02-01

Ce document présente l'installation de Chimère. Il a été mis à jour pour la version 2.0.0 de Chimère.

Votre session doit être initialisée avec ces variables d'environnement en ligne de commande :

```
CHIMERE_PATH=/srv/chimere # changez avec votre répertoire d'installation
CHIMERE_LOCALNAME=mychimere # changez avec le nom de votre projet
CHIMERE_APP_PATH=$CHIMERE_PATH/$CHIMERE_LOCALNAME
```

Une fois l'application installée, il y a un certain nombre d'étapes à suivre pour configurer *votre* Chimère.

La plupart de ces étapes sont faites dans les pages web d'administration.

Si vous n'êtes pas familiarisé avec les pages d'administration de *type Django* vous pouvez dès maintenant regarder le premier paragraphe de l'*Administration* où elles sont présentées.

Pour accéder à ces pages vous avez à vous identifier avec un compte ayant pour état *équipe* ou *super-utilisateur*.

Un compte *super-utilisateur* est créé à l'initialisation de la base de données.

3.1 Gérer les zones

Une zone est la base de votre carte. Pour une zone il est défini :

- un nom : un libellé pour cette zone ;
- une URN associée (*facultatif*) : le nom de la zone en tant que ressource Web. En pratique si la zone définie n'est pas celle par défaut, elle est utilisée à la fin de l'adresse Web de base pour pouvoir accéder à cette zone. Ce n'est pas obligatoire mais nécessaire en pratique pour chaque zone qui n'est pas celle par défaut ;
- un message par défaut (*facultatif*) : ce message est affiché une fois par jour par utilisateur consultant la carte ;
- un ordre (pour trier les zones) ;
- une disponibilité ;
- un état « *par défaut* ». La zone *par défaut* est vue par défaut. Une seul zone peut être *par défaut* : activez cet état sur une zone le désactive sur toutes les autres ;
- des catégories cochées par défaut (*facultatif*) ;
- si les catégories sont affichées dynamiquement. Si les catégories sont affichées dynamiquement, l'utilisateur ne voit seulement que les catégories qui ont des éléments sur la portion de carte actuellement à l'écran ;
- des restrictions sur les catégories (*facultatif*) : si aucune restriction n'est définie, toutes les catégories sont disponibles ;

— une feuille de style CSS externe (*facultatif*) : une adresse Web qui pointe vers une feuille de style CSS externe ;
— une restriction à la portion de carte : si coché, l'utilisateur ne pourra pas faire glisser la carte en dehors de la portion de carte. À cause de limitations de la bibliothèque OpenLayers utilisée par Chimère, il n'y a pas de restriction sur le zoom ;
— une portion de carte : c'est la zone qui sera affichée par défaut en arrivant sur la carte. Si la restriction sur une portion de carte est activée, la restriction portera sur cette portion. Laissez appuyée la touche *Control*, cliquez et glissez pour dessiner la portion de carte choisie.
— calques disponibles (*facultatif* : OSM Mapnik est utilisé par défaut) : les rendus OSM Mapnik, OSM MapQuest, OSM Transport Map, OSM CycleMap sont disponibles par défaut. Vous pouvez ajouter de nouveaux calques (cf. *Gestion des calques*).

Les *Zones* sont personnalisables directement depuis l'interface d'administration dans *Chimere* > *Zones*.

Comme il y a peu de chance que la zone définie par défaut vous convienne, il sera au minimum nécessaire de définir une zone par défaut.

Ajouter plusieurs zones peut être un moyen d'afficher vos données de différentes manières.

3.2 Gestion des utilisateurs

Si vous n'êtes pas le seul administrateur/modérateur de cette installation de Chimère vous aurez à créer et gérer des comptes pour les autres utilisateurs.

Vous pouvez créer un nouvel administrateur en ligne de commande :

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

Les mots de passe peuvent être changés en ligne de commande (utile si vous avez oublié votre mot de passe) :

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

Les *Utilisateurs* sont directement éditables depuis les pages d'administration au niveau de la section *Auth/Utilisateur*.

Pour créer un nouveau compte, cliquez simplement sur le bouton *Ajouter* à côté de *Utilisateur*. Donnez un nom et un mot de passe (l'utilisateur pourra changer son mot de passe plus tard).

Ensuite complétez les autres informations.

Cochez la case : *Statut équipe* (ou cet utilisateur ne sera pas capable d'accéder aux pages d'administration).

Si ce compte est un nouvel administrateur technique, cochez la case *Statut superutilisateur* (cet utilisateur doit être digne de confiance !). Sinon vous allez devoir donner des permissions à ce nouvel utilisateur. Plutôt que d'assigner manuellement des permissions aux utilisateurs, il est plus simple de leur affecter un groupe avec des permissions pré-définies.

Deux types de groupe sont proposés par défaut : les administrateurs de l'application et les modérateurs.

Les groupes de modérateurs ont des droits limités à une seule zone (le nom du groupe est *Nom_de_zone modération*). Ils ne voient que les éléments qui concernent leur zone. Un utilisateur pouvant faire partie de plusieurs groupes, il peut modérer plusieurs zones.

Détails des droits pour les groupes par défaut :

Élément (ajout/modification/suppression)	Administrateur technique	Administrateur de l'application	Modérateur
Utilisateur	Oui	Non	Non
Groupe	Oui	Non	Non
Modèle de propriété	Oui	Non	Non
Import	Oui	Non	Non
Calque	Oui	Non	Non
Nouvelles	Oui	Oui	Non
Zone	Oui	Oui	Non
Icône	Oui	Oui	Non
Couleurs/thème de couleur	Oui	Oui	Non
Catégorie/Sous-catégorie	Oui	Oui	Non
Point d'intérêt	Oui	Oui	Oui
Trajet	Oui	Oui	Oui

3.3 Créer des modèles de propriété

Une installation de base de Chimère permet d'associer un nom, des catégories, une description, des dates, des fichiers multimédias, des fichiers d'image à chaque élément géographique.

Vous souhaitez peut-être des champs personnalisés tels que des numéros de téléphone ou des horaires d'ouverture. Pour cela, il suffit d'ajouter un nouveau modèle de propriété (*Chimere/Modèle de propriété*).

La page d'administration vous demande :

- un nom ;
- un ordre (pour ordonner les propriétés entre elles) ;
- une disponibilité pour l'utilisateur (cela peut être utilisé pour associer des propriétés cachées) ;
- un état « Obligatoire » qui oblige à remplir ce champ dans les formulaires ;
- les catégories auxquelles associer cette propriété (si aucune catégorie n'est sélectionnée, la propriété est disponible pour toutes les categories) ;
- le type : texte, texte long, mot de passe ou date.

Warning : Pour rendre cette propriété disponible, il est nécessaire de recharger le serveur Web (les propriétés sont mis en cache).

Les formulaires sont alors automatiquement mis à jour avec ce nouveau champ.

En tant qu'administrateur, si vous ne souhaitez pas rendre disponible l'ajout ou la modification des propriétés, vous pouvez désactiver la gestion des modèles de propriété en mettant *CHIMERE_HIDE_PROPERTYMODEL* à la valeur *True* dans votre fichier *local_settings.py*.

CHAPITRE 4

Administration

Auteur Étienne Loks
date 2012-11-29

Ce document présente l'administration de Chimère. Il a été mis à jour pour la version 2.0.0 de Chimère.

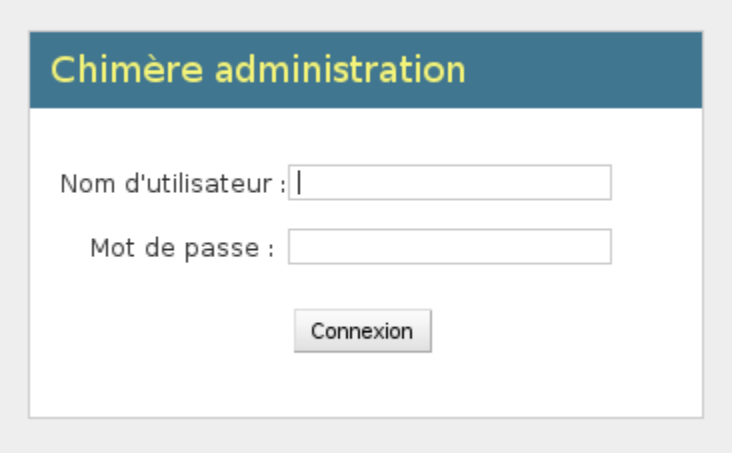
4.1 Présentation des pages d'administration

Les pages d'administration sont accessibles à l'adresse : http://where_is_your_chimere/admin/

N'oubliez pas la barre oblique (slash) à la fin de l'adresse.

4.1.1 Identification

Tout d'abord vous avez à vous identifier avec l'identifiant et le mot de passe fournis.



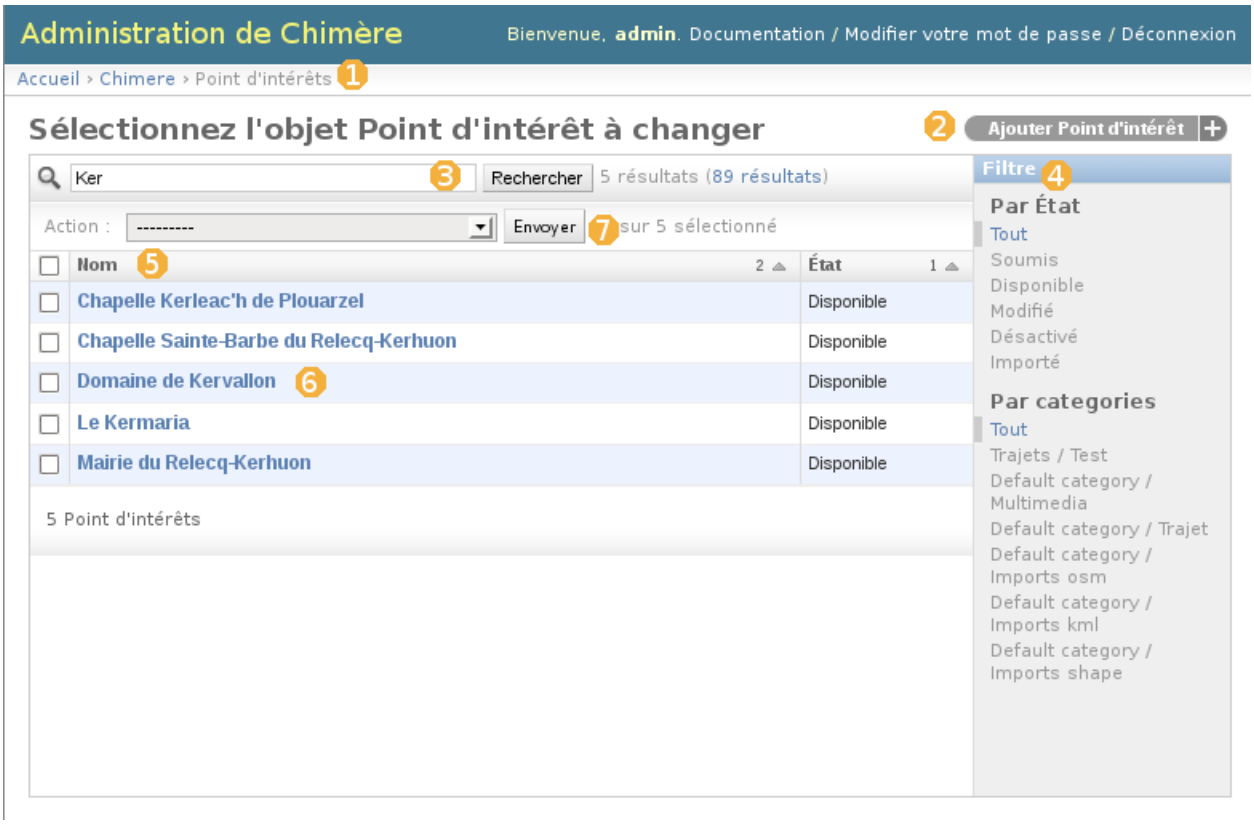
4.1.2 Page principale

Une fois identifié, vous avez accès à la page principale d'administration.

Cela s'affiche ainsi :

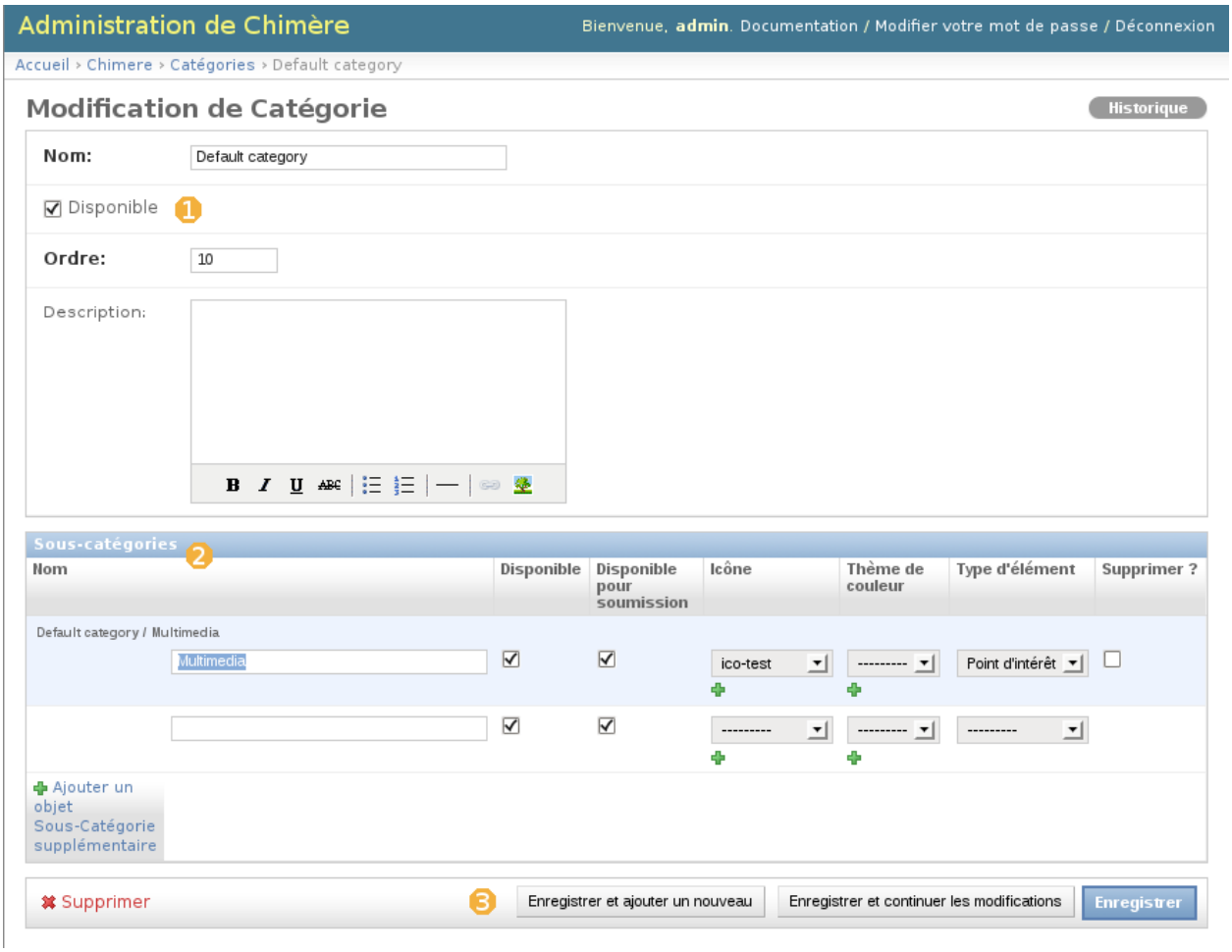
1. lien vers cette **Documentation**, vers le formulaire de **Changement de mot de passe** et la déconnexion ;
2. la liste des actions récemment faites avec ce compte ;
3. un titre d'application, la plupart des actions vont se faire dans l'application **Chimere** ;
4. un élément à l'intérieur de l'application. Depuis ces pages, vous pouvez *Ajouter* un nouvel élément ou consulter/**Changer** des éléments. Le lien **Ajouter** conduit au Formulaire des éléments. Le lien **Modifier** conduit à la Liste des éléments. La Liste des éléments est également disponible en cliquant sur le libellé de l'élément.

4.1.3 Liste des éléments



1. chemin dans le site d'administration. C'est un raccourci pratique pour revenir à la page principale.
2. lien pour créer un nouvel élément depuis la liste des éléments.
3. recherche des éléments par mot (n'est pas disponible pour tous les types d'éléments).
4. cette boite permet de filtrer les entrées actuelles avec des filtres (n'est pas disponible pour tous les types d'éléments)
5. les en-têtes de cette table sont cliquables. Cliquer sur une en-tête permet de trier les éléments de manière ascendante et descendante. Un tri multi-en-tête est possible (le nombre à droite de l'en-tête indique l'ordre de prise en compte dans le tri).
6. chaque élément peut être coché (pour lui appliquer une action) ou sélectionné (en cliquant sur la première colonne) pour voir son détail et éventuellement le modifier ou le supprimer.

4.1.4 Formulaire des éléments



1. les champs pour l'élément sélectionné (ou vide si c'est un nouvel élément) sont affichés dans ce formulaire. Parfois certains champs sont en lecture seule et d'autres sont cachés. Les champs obligatoires ont leur intitulé en gras. Les changements sur ces champs ne sont effectifs qu'une fois le formulaire validé.
2. pour certains éléments il y a des sous-éléments associés. Ces sous-éléments peuvent être modifiés directement dans ce formulaire. Lorsque plusieurs sous-éléments sont associés à un élément, ils peuvent être réagencés par glisser-déposer.
3. le formulaire doit être validé par un de ces boutons. Ils parlent d'eux-même.

4.1.5 États

Les *États* sont des propriétés rattachées à chaque élément géographique dans Chimère. Pour administrer Chimère efficacement il est nécessaire de comprendre chacun de ces états.

— **Proposé** : État d'un élément nouvellement proposé par un utilisateur. Cet élément n'est pas visible sur la carte.
— **Disponible** : État d'un élément visible sur la carte.
— **Désactivé** : État d'un élément écarté.
— **Modifié** : État d'une proposition de modification d'un élément par un utilisateur.
— **Importé** : État d'un élément nouvellement importé. Les opérations d'import et d'export nécessitent que tous les éléments avec l'état *importé* soient traités (validés, désactivés ou supprimés).

4.2 Gestion des nouvelles

Un système de nouvelles est disponible. Tout ce que vous avez à faire est de cliquer sur le bouton *Ajouter* à côté de *Nouvelles*. Pour chaque nouvelle il est nécessaire de fournir un nom et un contenu. Le contenu peut contenir des balises HTML. La disponibilité est gérée avec une case à cocher.

4.3 Création de catégories/sous-catégories

Avant l'ajout de catégories, il est nécessaire de définir des icônes. Ces icônes apparaissent sur la carte et sur la boîte contenant les catégories sur la carte principale. Faites attention de bien redimensionner vos icônes. En effet les icônes vont être présentées à leur taille réelle sur la carte. Pour ajouter des icônes : cliquez sur le bouton **Ajout** à côté de *Icônes*.

Le site http://mapicons.nicolasmollet.com/ permet de générer facilement des icônes adaptées à un usage dans Chimère.

Les catégories sont en fait des conteneurs à sous-catégories. Il est juste nécessaire de fournir nom et ordre d'affichage. Pour ajouter des catégories : cliquez sur le bouton **Ajout** près des catégories.

Les champs concernant les sous-catégories sont : un nom, une icône, un ordre, un thème de couleur et un type d'élément. La plupart des champs parlent d'eux-mêmes. Les thèmes de couleurs sont composés de plusieurs couleurs. Les couleurs sont utilisées pour le tracé des trajets (si la sous-catégorie contient des trajets). Si c'est une couleur de base, cela peut être défini par le nom en anglais (par exemple *red* pour rouge, *blue* pour bleu, *purple* pour violet) sinon vous pouvez donner le code couleur HTML RVB (par exemple *#9227c9*). Le type d'élément est le type d'élément que la sous-catégorie peut contenir : points d'intérêts, trajets ou les deux.

4.4 Édition/modération des éléments

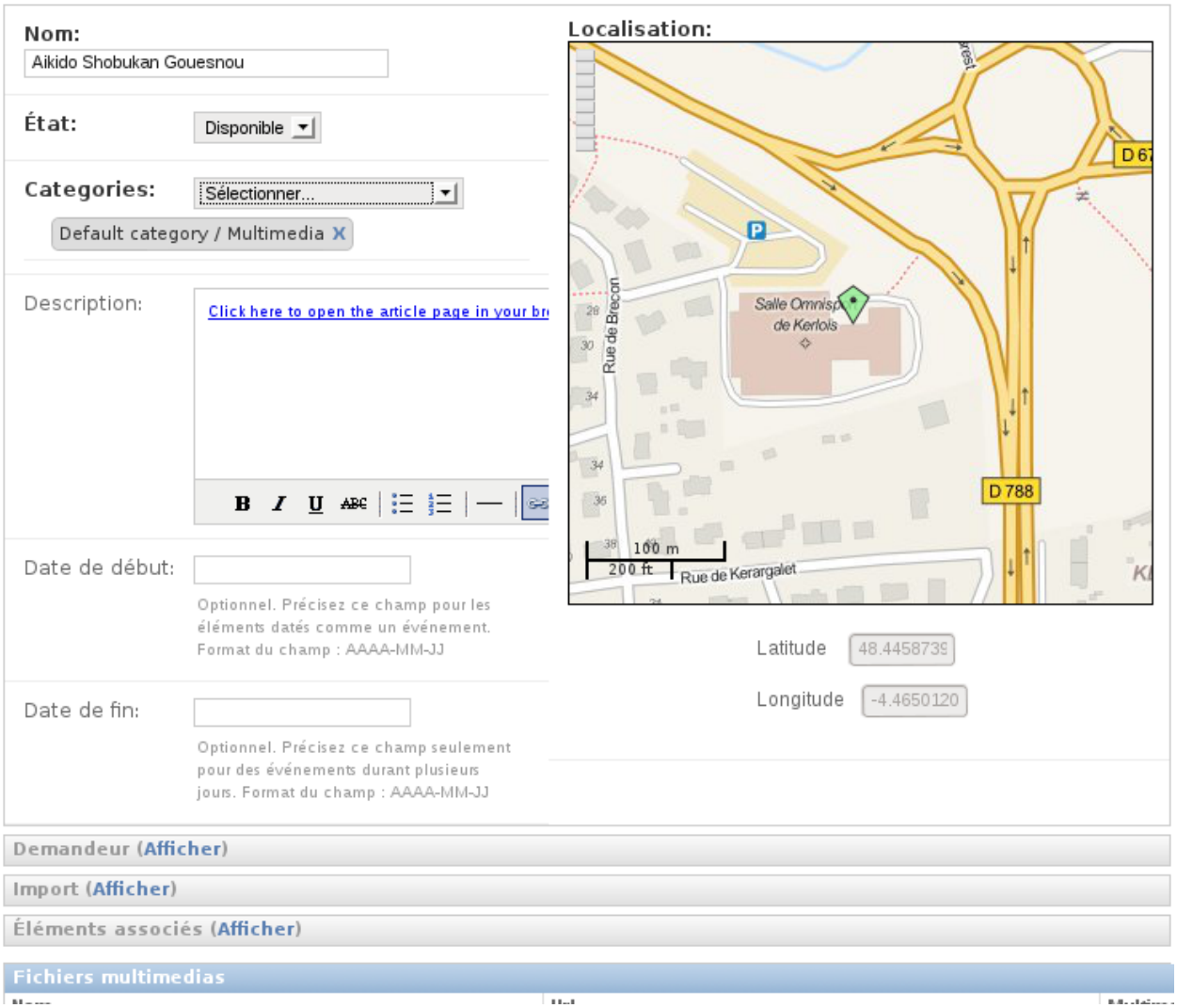
L'étape de modération est relativement simple. Elle fonctionne de la même manière avec les points d'intérêt ou avec les trajets. Le modérateur accède classiquement aux points d'intérêts (ou trajets) en cliquant sur leur nom dans la liste d'éléments.

Un champ de recherche est disponible pour rechercher par nom mais il est généralement plus intéressant de filtrer par état et sous-catégories.

Il y a un certain nombre d'action disponible.

- **Supprimer** pour supprimer les éléments sélectionnés. Une étape de confirmation est affichée.
- **Valider** pour donner le status *Disponible* aux éléments sélectionnés.
- **Désactiver** pour donner le status *Désactivé* aux éléments sélectionnés. C'est particulièrement utile pour garder des éléments que vous ne voulez pas voir apparaître sur la carte mais conserver en base de données.
- **Gérer les éléments modifiés** pour gérer les propositions de modification par les utilisateurs sur le site principal (cf. *Gérer les modifications des utilisateurs/les éléments importés ayant des modifications locales*). Les éléments modifiés ne peuvent être traités qu'un par un.
- **Export en...** pour exporter les éléments sélectionnés vers le format sélectionné.

Pour modifier un élément, classiquement, vous cliquez sur son nom pour accéder ensuite à un formulaire pour modifier librement l'élément.



Sur ce formulaire il y a tous les éléments disponibles à l'utilisateur plus quelques champs supplémentaires.

- Les champs *Imports* ne sont pertinent que pour les données importées depuis une source externe ou pour les données destinées à être exportées vers OSM cf. à la *section import* de cette documentation.
- Les *Éléments associés* sont des champs en lecture seule qui listent les éléments associés à l'élément courant (élément de référence d'une modification, fichier associé à un trajet).

Les éléments multimédias sont listés à la fin du formulaire. Vous pouvez librement ajouter, modifier, changer l'ordre (avec du glisser-déposer) de ces éléments.

Si un élément n'est pas pertinent, le bouton **Supprimer** permet de le supprimer.

> **Warning :** N'oubliez pas de valider vis changements avec un des boutons d'enregistrement disponibles à la fin du formulaire (notamment il est assez facile d'oublier de confirmer les changements faits aux éléments multimédias).

4.5 Gérer les modifications des utilisateurs/les éléments importés ayant des modifications locales

Des propositions de modification peuvent être faits sur le site principal par les utilisateurs.

Dans Chimère, une proposition de modification est un nouvel élément avec l'état **Modifié** qui dispose d'un lien vers l'élément de référence.

Vous pouvez avoir aussi des éléments importés qui ont à la fois des modifications locales et sur la source externe. La nouvelle version de la source externe a l'état **Importé** et un lien vers l'élément de référence.

Note : Si vous êtes identifié en tant qu'administrateur et que vous faites des changements sur la carte avec le « formulaire utilisateur » les changements vont être directement pris en compte.

Un formulaire spécifique a été développé pour faciliter le traitement de ces éléments modifiés.

Vous pouvez accéder à ce formulaire spécifique avec l'action *Gérer les éléments modifiés*.

Ce formulaire est un tableau à 3 colonnes.

1. La première colonne affiche les informations de l'élément de référence.
2. La seconde colonne affiche les informations que propose l'utilisateur.
3. La troisième colonne est une liste de cases à cocher. Après validation, pour chaque ligne cochée, la valeur de l'élément modifié remplacera la valeur de l'élément de référence.

Note : Pour rejeter toutes les modifications proposées, validez le formulaire sans cocher aucune case.

CHAPITRE 5

Import/export

Author Étienne Loks

date 2013-02-01

Ce document présente les fonctions d'import et d'export de Chimère. Ce document a été mis à jour pour la version 2.0.0 de Chimère.

5.1 Import

Dans Chimère, le mécanisme d'import est basé sur les objets **Import**. Ces objets sont stockés dans une base de données pour garder trace des imports et pour faciliter la ré-importation depuis une même source. En fait, si cela est possible, la mise à jour de données depuis un même type de source est gérée, de préférence à une ré-importation.

Note : La possibilité de réaliser de telles mises à jour est conditionnée à l'existence d'un identifiant unique pour chaque objet de la source.

Pour ajouter un objet **Import**, vous devez aller dans *Chimere* > *Imports* puis cliquer sur **Ajouter**.

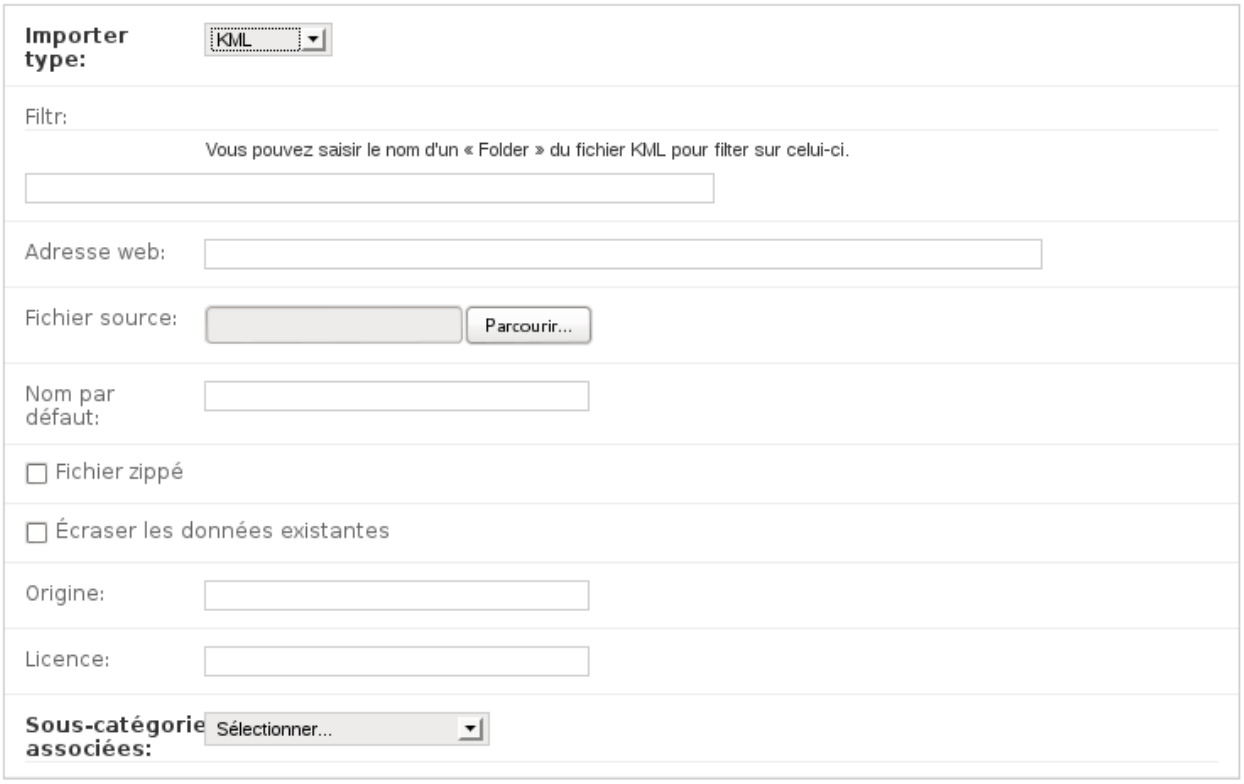
Après cela, vous aurez à sélectionner votre type de source. Le formulaire suivant dépend de ce type de source.

5.1.1 Champs communs à tous les types de source

- **Nom par défaut** : si aucun nom ne peut être trouvé dans la source pour ce nouvel objet le nom par défaut sera utilisé. Si ce champ est vide le nom de la catégorie associée sera utilisée.
- **SRID** : Chimère tente d'identifier automatiquement le système de coordonnées utilisé par la source. Mais parfois l'information n'est pas présente ou ne peut pas être devinée (par exemple un fichier Shapefile qui utilise un fichier proj non standard). Dans ce cas, Chimère utilise WGS84 par défaut (latitude et longitude). Si vous avez des problèmes avec la localisation des éléments vous devez probablement mettre ici le SRID correspondant au système de coordonnées de votre source.
- **Écraser les données existantes** : par défaut quand les données ont été mises à jour à la fois sur la source externe et sur votre source externe un nouvel élément est créé et a à être rapproché avec le *formulaire de gestion des modifications*. Si vous ne souhaitez pas avoir à faire ce rapprochement et alors écraser les données existantes avec les données de la source externe, cochez cette option.
- **Obtenir la description depuis la source** : si cette case est cochée, l'importeur va essayer d'obtenir une description depuis le fichier source. Cette option n'est valable que pour certains formats.
- **Description par défaut** : Une description par défaut à ajouter aux nouveaux éléments. Ce champ n'est disponible que lorsque **Obtenir la description depuis la source** n'est pas coché.

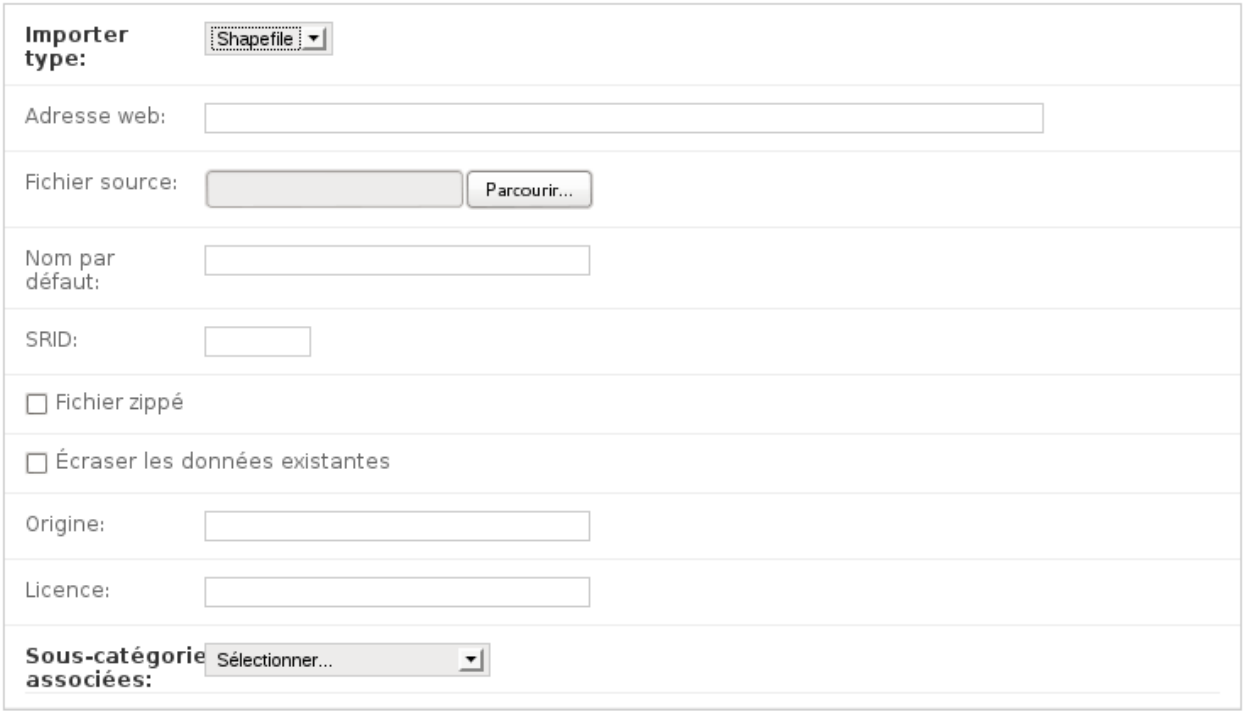
— **Origine** : si non nul, ce champ va être associé à chaque élément importé afin d'identifier facilement d'où l'élément provient. Pour les imports OSM la source est ajoutée automatiquement.
— **Licence** : si non nul, ce champ va être associé à chaque élément importé afin d'identifier facilement la licence de l'élément. Pour les imports OSM la licence est ajoutée automatiquement.
— **Sous-catégories (obligatoire)** : les sous-catégories sélectionnées seront associées automatiquement aux nouveaux éléments importés.

5.1.2 Import KML



— **Adresse Web / fichier source (obligatoire)** : votre fichier KML peut être local ou distant. Vous avez à remplir un des deux champs.
— **Filtre** : si vous souhaitez importer seulement un dossier (**Folder**) du fichier KML mettez son nom dans ce champ.
— **Fichier zippé** : si votre source est un fichier KMLZ (un fichier KML zippé), cochez cette case.

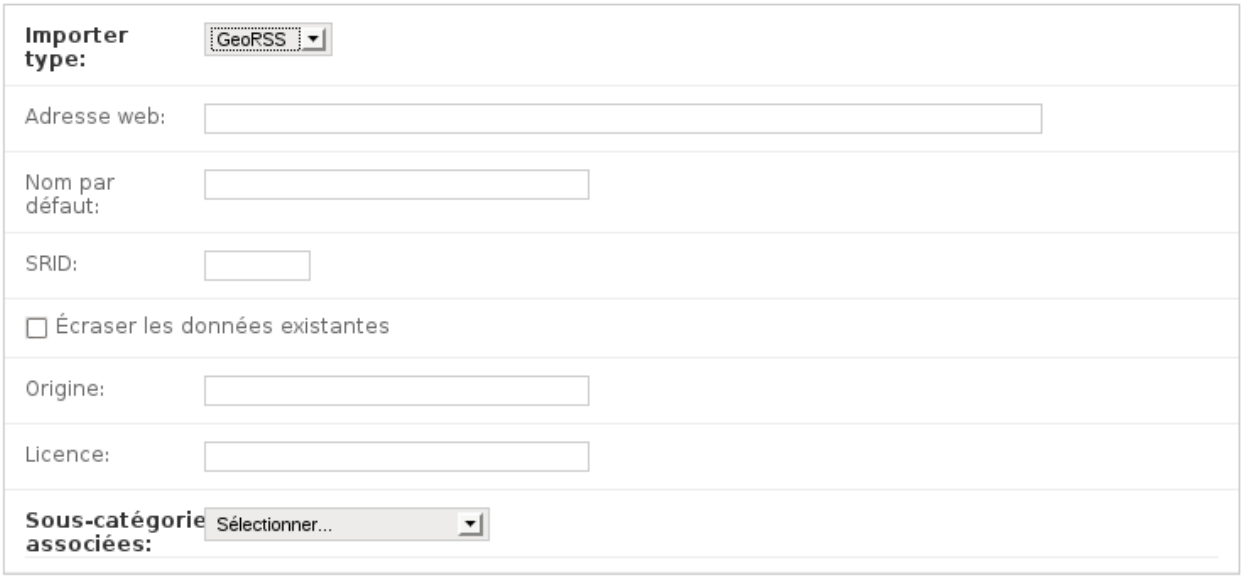
5.1.3 Import Shapefile



— **Adresse Web / fichier source (obligatoire)** : votre fichier shapefile peut être local ou distant. Vous avez à remplir un des deux champs.
— **Fichier zippé** : seuls les fichiers shapefile zippés sont acceptés aussi cette case devrait être cochée.

5.1.4 Import GeoRSS

Simple GeoRSS et W3C GeoRSS sont gérés.



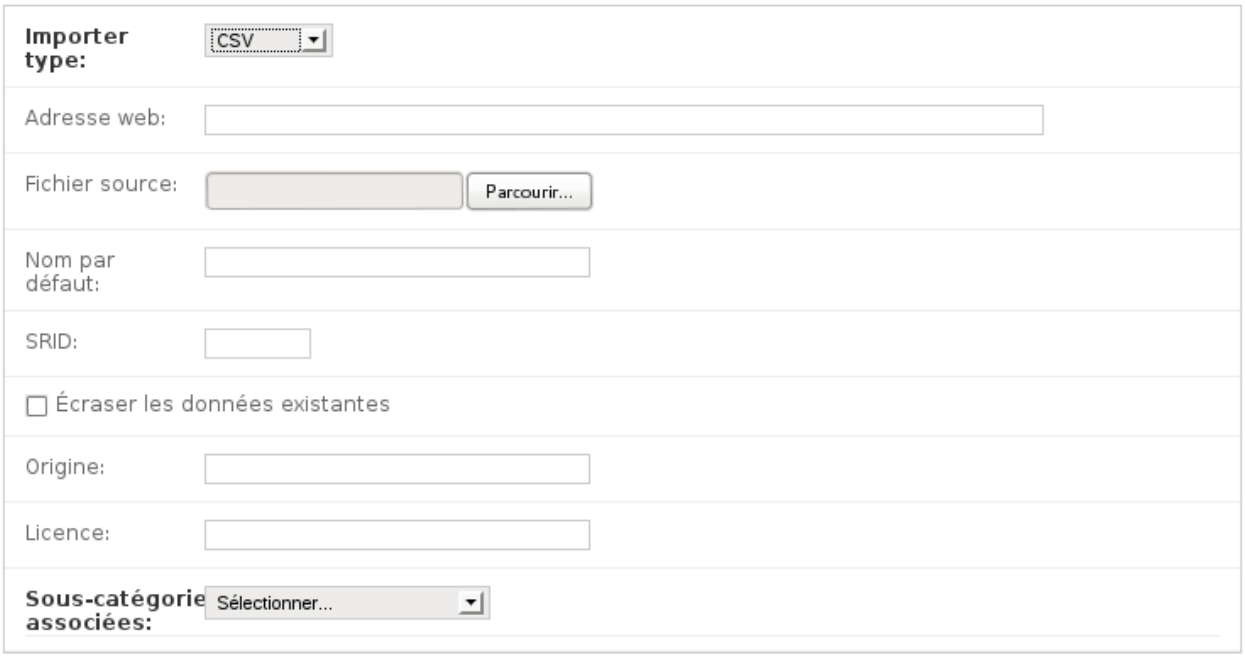
— **Adresse Web (obligatoire)** : seul les flux GeoRSS distant sont gérés.

5.1.5 Import CSV

Le format du fichier CSV (nombre et ordres des colonnes) géré par Chimère varie en fonction des modèles de propriété que vous avez utilisé sur votre instance Chimère. Aussi, il est recommandé dans un premier temps de faire un export CSV de quelques éléments. Le format du fichier CSV exporté sera compatible avec Chimère pour l'import.

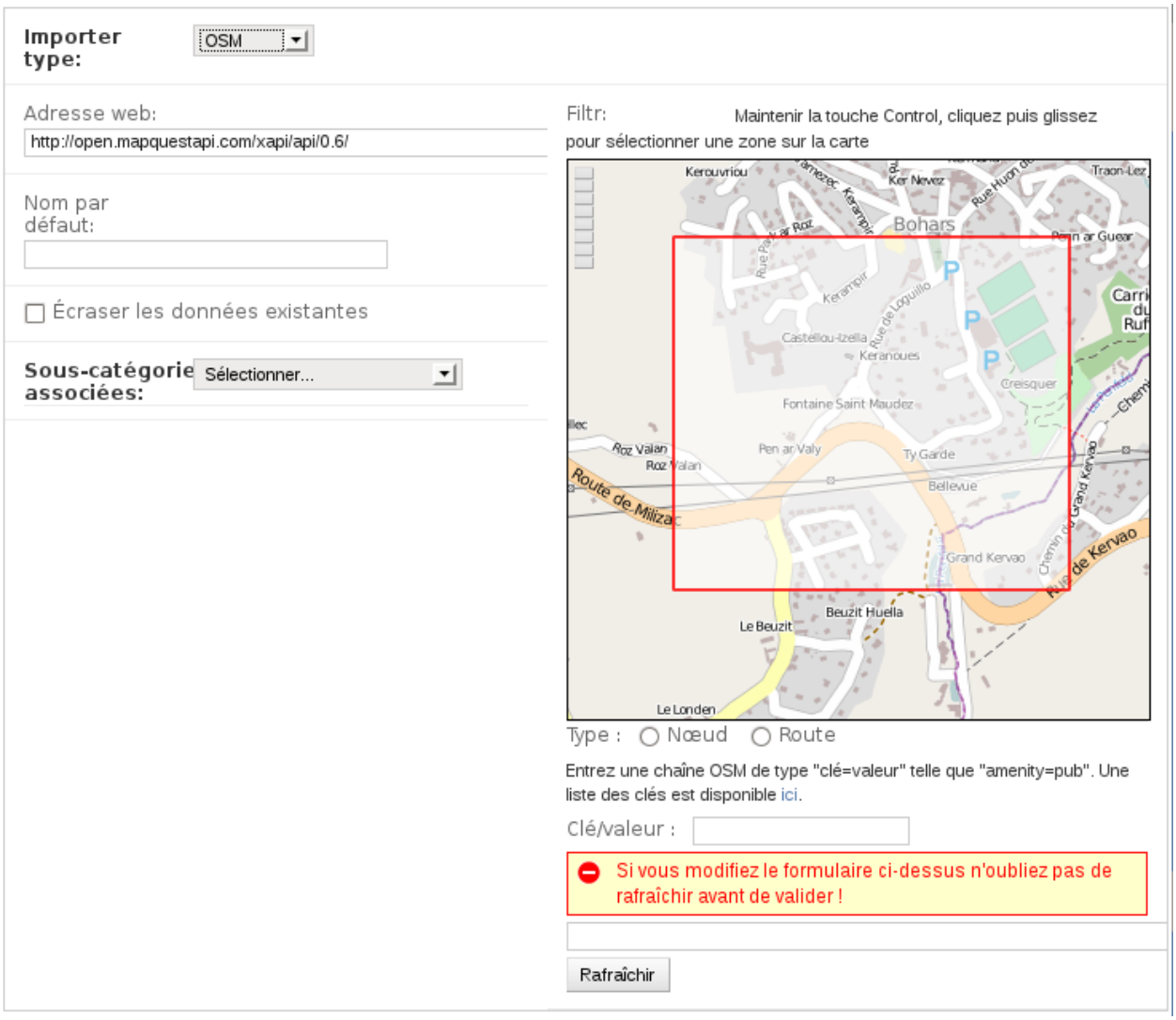
En tout cas à cause des champs géographiques ce format n'est pas très pratique pour l'ajout de nouveau contenu mais peut s'avérer utile pour les mises à jour d'information.

Warning : Si vous souhaitez mettre à jour des données existantes avec cet import, à moins que vous sachiez éditer du WKT ne modifiez **pas** la colonne qui concerne la géométrie de l'élément.



— **Adresse Web/fichier source (obligatoire)** : votre fichier CSV peut être distant ou local. Vous avez à remplir un des deux champs.

5.1.6 Import OpenStreetMap



Pour importer depuis OSM, Chimère utilise l'API XAPI d'OSM.

- **Adresse Web (obligatoire)** : l'URL XAPI à utiliser pour importer. Ce champ doit être rempli par défaut. Par défaut le serveur MapQuest est utilisé car il semble le plus robuste. Si vous avez des problèmes avec l'import de données OSM, vérifiez la disponibilité du serveur utilisé et le cas échéant changez le.
- **Filtre sur zone (obligatoire)** : dessinez la section de carte à utiliser pour votre import OSM.
- **Filtre sur types (obligatoire)** : choisissez si vous souhaitez importer des routes ou des nœuds.
- **Filtre sur les clé/valeur (obligatoire)** : choisissez la paire clé/valeur à utiliser pour filtrer les données OSM. Un lien vers la page de « Map features » OSM est fourni pour vous aider à trouver les valeurs adaptées.
- **Bouton de rafraîchissement** : ce bouton convertit vos choix en arguments XAPI adaptés. N'oubliez pas de presser sur ce bouton avant de valider le formulaire.

5.1.7 Importer

Une fois que le nouvel objet *Import* est créé, sélectionnez le dans la liste des objets, choisissez *Importer* et valider.

L'import doit se dérouler normalement. Dans le cas contraire, un message d'erreur explicite doit s'afficher dans la colonne *État* de votre import.

Vous pouvez aussi lancer vos imports en ligne de commande (idéal pour les travaux à mettre dans la table *cron*). Dans le répertoire du projet, il est juste nécessaire de lancer la commande

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

— *import_id* est l'identifiant de l'import

Si vous lancez l'import en ligne de commande sans l'identifiant d'import, la liste des imports disponibles est affichée et vous pouvez alors en choisir un.

5.1.8 Gérer les données importées

Tous les nouveaux éléments importés ont l'état **Importé**. Pour que ceux-ci soient disponible sur la carte, il est nécessaire de les valider. Si vous ne souhaitez pas afficher certains éléments plutôt que de les supprimer, il est recommandé de les mettre à l'état **Désactivé**. Ainsi lors de la prochaine mise à jour depuis la source, ceux-ci resteront désactivés plutôt que d'apparaître comme nouveaux éléments.

Warning : Soyez vigilants avec les doublons entre les données existantes et les données importées. C'est particulièrement important si vous souhaitez exporter vos données vers OSM.

5.2 Export

5.2.1 Exporter vers CSV/KML/Shapefile

Depuis les *listes d'éléments géographiques* vous pouvez exporter directement vers le format choisi. Tout ce que vous avez à faire c'est de sélectionner les éléments que vous souhaitez exporter, choisir l'action appropriée dans la liste d'action et de valider.

Vous pouvez aussi lancer les exports depuis la ligne de commande (idéal pour les travaux à mettre dans la table *cron*). Dans le répertoire du projet, vous avez juste à lancer

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

— *subcategory_id* est l'identifiant de la sous-categorie choisie ;
— *CSV|KML|SHP* est le format choisi ;
— *marker|route* est pour obtenir points d'intérêts (marker) ou trajets (route) ;
— *filename* est le nom du fichier de sortie

Si vous lancez la commande sans arguments il vous sera demandé les choix à faire pour votre export.

5.2.2 Exporter vers OSM

Warning : Si vous n'êtes pas sûr de ce que vous êtes entrain de faire avec les exports vers OSM : **ne le faites pas !** C'est vraiment important de ne pas plaisanter avec les données des autres.

Note : Seuls les exports des nœuds OSM sont gérés.

Les exports ne sont pas aussi facile à gérer que les autres exports. Tout d'abord (si cela n'est pas déjà fait) vous avez à définir un import OSM (*regarder dessus* pour plus de détail). Cela permettra de déterminer :

— la zone géographique concernée par votre export ;
— la clé/valeur à ajouter à vos éléments (nouveaux ou mis à jour) ;
— les sous-catégories concernées par cet export. Si vous pensez que certains éléments dans ces sous-catégories ne devraient pas être dans la base de données OSM (car ils ne sont pas pertinents ou à cause de question de licence) marquez les préalablement comme **À ne pas exporter vers OSM** dans les *champs d'imports* des *formulaires concernant les éléments géographiques*.

L'export vers OSM dans Chimère est fait de sorte à être le plus conservateur possible par rapport à la base de données OSM. C'est pour cela qu'avant tout export, un import est fait. Si le nouvel import a des données mises à jour, il est nécessaire de retraiter les nouvelles données importées avant de faire un export (cf. *gérer les données importées*).

Pour lancer un export sélectionnez l'objet *Import* approprié dans la liste des imports. Ensuite sélectionnez l'action **Exporter vers OSM** et validez. Puis on vous demande votre identifiant OSM, votre mot de passe OSM et l'API que vous souhaitez utiliser. Si vous comptez faire des exports régulièrement avec Chimère, il est recommandé de créer un compte spécifique pour cela. L'API de test est disponible pour faire des tests d'export. Si vous souhaitez utiliser l'API de test, vous aurez à créer un compte spécifique sur la plateforme de test.

Warning : Les données sur la plateforme de test ne sont pas synchronisées avec la plateforme principale. Vous n'aurez pas les mêmes données que celles celles importées avec XAPI.

Une fois que tous ces champs sont remplis, vous pouvez (enfin !) lancer l'export.

Quand vous exportez, des couples clés/valeurs sont automatiquement ajoutés/mis à jour dans la base de données OSM :
— *name* : obtenu depuis le nom de l'élément dans Chimère ;
— *source* : pour identifier Chimère comme une source.

CHAPITRE 6

Personnalisation

Auteur Étienne Loks

date 2012-11-29

Ce document présente la personnalisation de Chimère. Ce document a été mis à jour pour la version 2.0.0 de Chimère.

6.1 Gestion des calques

Il y a différents calques disponibles par défaut dans Chimère (OSM Mapnik, OSM Mapquest, OSM Transport map, OSM Cyclemap). Vous pouvez ajouter d'autres calques en utilisant les pages d'administration de Chimère.

Le nouveau calque est défini en utilisant une chaîne de code Javascript adéquate de la bibliothèque Openlayers. Ce code Javascript doit être une instance de *Openlayers Layer*, sans point virgule final.

Par exemple définir un calque Bing peut être fait avec un code de ce type :

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

Référez vous à la documentation de l'API Openlayers pour plus de détail.

6.2 Personnaliser l'agencement et le design

Si vous souhaitez simplement améliorer la feuille de style CSS, le plus simple est d'ajouter un lien vers une feuille de style supplémentaire dans vos *Zones* (cf. *Gérer les zones*).

Si vous souhaitez faire des changements plus importants dans l'agencement et la présentation, le projet *example_project* peut être personnalisé pour correspondre à vos besoins. Chaque fichier de patron de page présent dans le dossier *chimere/templates* peut être copié dans votre dossier *monprojet/templates* puis modifié.

Il est juste nécessaire de copier les fichiers que vous souhaitez modifier. Ces fichiers sont écrits dans le langage de patron Django principalement composé de HTML avec des éléments de logique. Référez vous à la documentation des patrons Django pour plus de détails.