Moneta's Documentation

Release 1.8.11

Matthieu Gallet

Aug 01, 2018

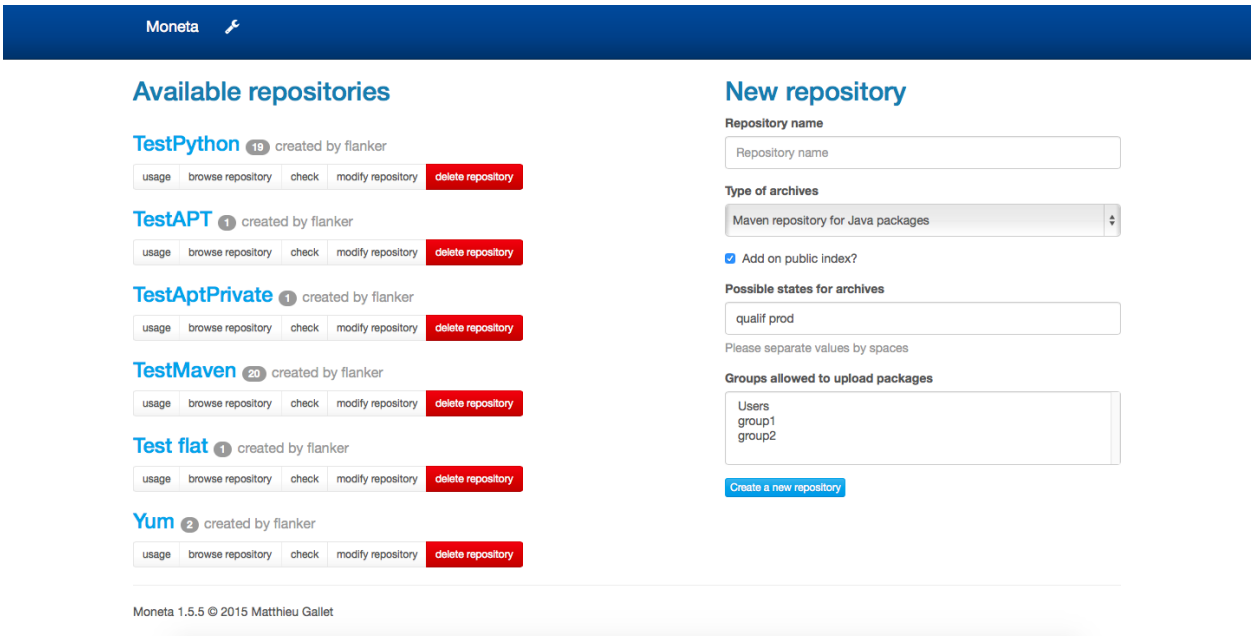
Contents

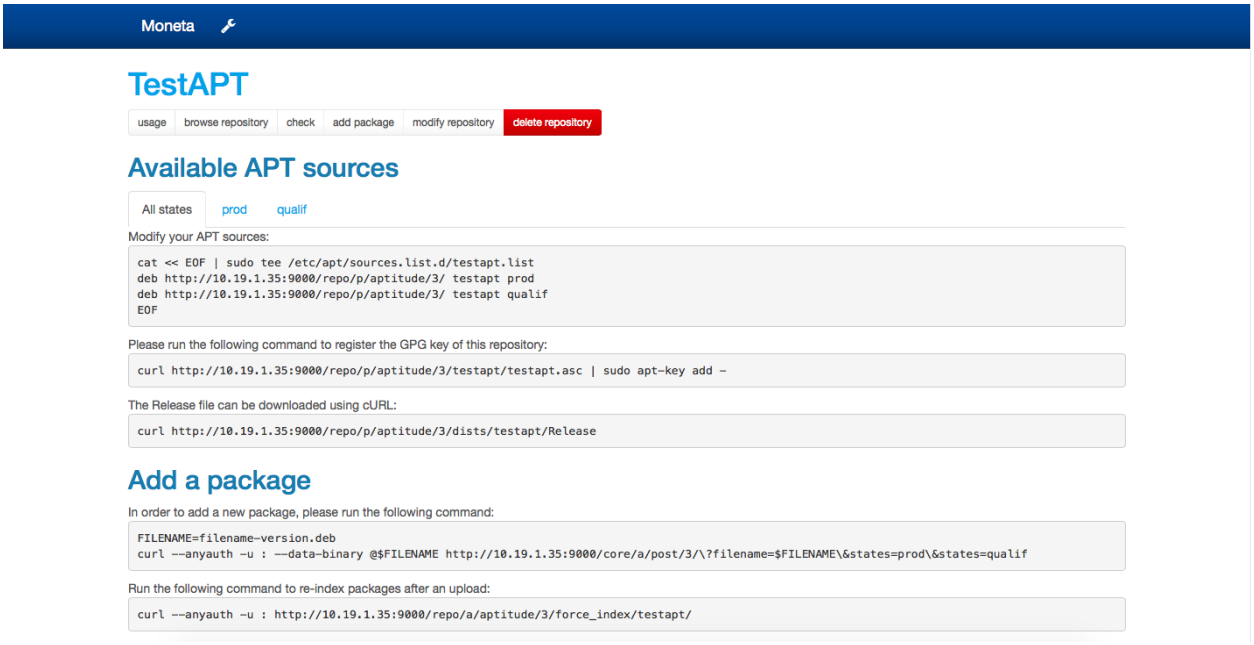
Moneta is a web application emulating different kinds of package repositories. The goal is not to replace official mirrors like packages.debian.org or mirror.centos.org, but to have a single location for all your Python/Java/Debian/RedHat private packages.

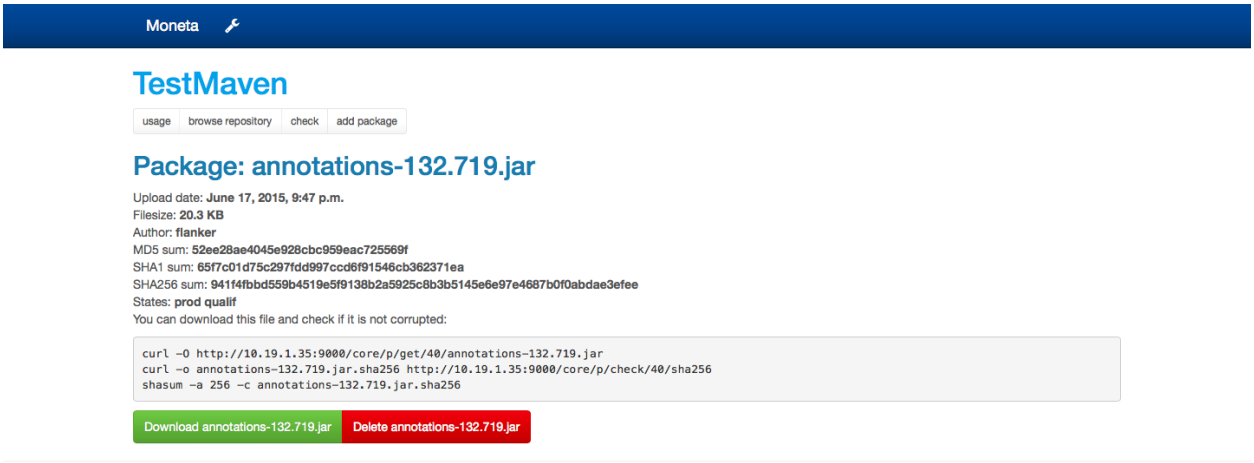
Repositories are created on-the-fly and you can limit upload rights to some groups of users.

Currently, you can create the following types of repositories:

- Aptitude for Ubuntu or Debian systems,
- Yum for CentOS, Fedora or Red Hat systems,
- Maven for Java or Scala packages,
- Pypi for Python packages,
- Gem for Ruby packages,
- Jetbrains for IntelliJ/PyCharm/PhpStorm/RubyMine/AppCode/CLion plugins,
- Vagrant boxes,
- any binary, versionned files.




Moneta
TestAPT
usage
browse repository
check
add package
modify repository
delete repository
Available APT sources
All states
prod
qualif
Modify your APT sources:
cat << EOF | sudo tee /etc/apt/sources.list.d/testapt.list
deb http://10.19.1.35:9000/repo/p/aptitude/3/ testapt prod
deb http://10.19.1.35:9000/repo/p/aptitude/3/ testapt qualif
EOF
Please run the following command to register the GPG key of this repository:
curl http://10.19.1.35:9000/repo/p/aptitude/3/testapt/testapt.asc | sudo apt-key add -
The Release file can be downloaded using cURL:
curl http://10.19.1.35:9000/repo/p/aptitude/3/dists/testapt/Release
Add a package
In order to add a new package, please run the following command:
FILENAME=filename-version.deb
curl --anyauth -u : --data-binary @$FILENAME http://10.19.1.35:9000/core/a/post/3/\?filename=$FILENAME\&states=prod\&states=qualif
Run the following command to re-index packages after an upload:
curl --anyauth -u : http://10.19.1.35:9000/repo/a/aptitude/3/force_index/testapt/


Moneta
TestMaven
usage
browse repository
check
add package
Package: annotations-132.719.jar
Upload date: June 17, 2015, 9:47 p.m.
Filesize: 20.3 KB
Author: flanker
MD5 sum: 52ee28ae4045e928cbc959eac725569f
SHA1 sum: 65f7c01d75c297fdd997ccd6f91546cb362371ea
SHA256 sum: 941f4fbbd559b4519e5f9138b2a5925c8b3b5145e6e97e4687b0f0abdae3efee
States: prod qualif
You can download this file and check if it is not corrupted:
curl -O http://10.19.1.35:9000/core/p/get/40/annotations-132.719.jar
curl -o annotations-132.719.jar.sha256 http://10.19.1.35:9000/core/p/check/40/sha256
shasum -a 256 -c annotations-132.719.jar.sha256
Download annotations-132.719.jar
Delete annotations-132.719.jar

CHAPTER 1

table of contents

1.1 Quick installation

Moneta mainly requires Python (3.5, 3.6, 3.7).

You should create a dedicated virtualenvironment on your system to isolate Moneta. You can use pipenv or virtualenvwrapper.

For example, on Debian-based systems like Ubuntu:

```
sudo apt-get install python3.6 python3.6-dev build-essential
sudo apt-get install ruby  # required for Ruby mirrors
```

On VirtualBox, you may need to install rng-tools to generate enough entropy for GPG keys (otherwise the generation will be very slow):

```
sudo apt-get install rng-tools
echo "HRNGDEVICE=/dev/urandom" | sudo tee -a /etc/default/rng-tools
sudo service rng-tools restart
```

If these requirements are fullfilled, then you can gon on and install Moneta:

```
pip install moneta --user
moneta-ctl collectstatic --noinput  # prepare static files (CSS, JS, ...)
moneta-ctl migrate  # create the database (SQLite by default)
moneta-ctl createsuperuser  # create an admin user
moneta-ctl check  # everything should be ok
```

You can easily change the root location for all data (SQLite database, uploaded or temp files, static files, ...) by editing the configuration file.

```
CONFIG_FILENAME=`moneta-ctl config ini -v 2 | grep -m 1 ' - .ini file' | cut -d '"' -
↪f 2`
# prepare a limited configuration file
```

(continues on next page)

(continued from previous page)

```
mkdir -p `dirname $CONFIG_FILENAME`
cat << EOF > $CONFIG_FILENAME
[global]
data = $HOME/moneta
EOF
```

Of course, you must run again the *migrate* and *collectstatic* commands (or moving data to this new folder).

You can launch the server process:

```
moneta-ctl server
```

Then open http://localhost:8131 with your favorite browser.

You can install Moneta in your home (with the *–user* option), globally (without this option), or (preferably) inside a virtualenv.

1.2 Requirements

As said before, Python is required but this is not the only requirement:

- Python (3.5, 3.6, 3.7),
- setuptools>=1.0 Python package (automatically installed with Moneta),
- djangofloor>=1.0.25 Python package (automatically installed with Moneta),
- gnupg>=2.3 Python package (automatically installed with Moneta),
- rubymarshal Python package (automatically installed with Moneta),
- pyyaml Python package (automatically installed with Moneta),
- an optional Redis server for sessions and cache,
- mysqlclient (Python package) libmysqlclient and libmysqlclient-dev (system packages) if you want to use MySQL,
- psutil (Python package) to display system information on the monitoring page,
- psycopg2 (Python package), libpq and libpq-dev (system packages) if you want to use PostgreSQL,
- cx_Oracle (Python package) and the associated system packages if you want to use Oracle,
- django_redis (Python package) is you want to cache pages in Redis,
- django-allauth (Python package) for OAuth2 authentication,
- django-radius (Python package) for Radius authentication,
- django-auth-ldap (Python package) and libldap-dev (system package) for LDAP authentication,
- django_pam (Python package) for PAM authentication.

1.3 Installation

Here is a simple tutorial to install Moneta on a basic Debian/Linux installation. You should easily adapt it on a different Linux or Unix flavor.

Like many Python packages, you can use several methods to install Moneta. Of course you can install it from source, but the preferred way is to install it as a standard Python package, via pip.

1.3.1 Upgrading

If you want to upgrade an existing installation, just install the new version (with the *–upgrade* flag for *pip*) and run the *collectstatic* and *migrate* commands (for updating both static files and the database).

1.3.2 Ruby

If you want to use the Ruby mirror functionnality, Ruby is required on the server:

```
sudo apt-get install ruby
```

1.3.3 Preparing the environment

```
sudo adduser --disabled-password moneta
sudo chown moneta:www-data $DATA_ROOT
sudo apt-get install virtualenvwrapper python3.6 python3.6-dev build-essential␣
↪postgresql-client libpq-dev
sudo -u moneta -H -i
mkvirtualenv moneta -p `which python3.6`
workon moneta
```

1.3.4 Database

PostgreSQL is often a good choice for Django sites:

```
sudo apt-get install postgresql
echo "CREATE USER moneta" | sudo -u postgres psql -d postgres
echo "ALTER USER moneta WITH ENCRYPTED PASSWORD '5trongp4ssw0rd'" | sudo -u postgres␣
↪psql -d postgres
echo "ALTER ROLE moneta CREATEDB" | sudo -u postgres psql -d postgres
echo "CREATE DATABASE moneta OWNER moneta" | sudo -u postgres psql -d postgres
```

Moneta can use Redis for caching pages and storing sessions:

```
sudo apt-get install redis-server
```

1.3.5 Apache

Only the Apache installation is presented, but an installation behind nginx should be similar. Only the chosen server name (like *moneta.example.org*) can be used for accessing your site. For example, you cannot use its IP address.

```
SERVICE_NAME=moneta.example.org
sudo apt-get install apache2 libapache2-mod-xsendfile
sudo a2enmod headers proxy proxy_http xsendfile
sudo a2dissite 000-default.conf
# sudo a2dissite 000-default on Debian7
cat << EOF | sudo tee /etc/apache2/sites-available/moneta.conf
```

(continues on next page)

(continued from previous page)

```
<VirtualHost *:80>
    ServerName $SERVICE_NAME
    Alias /static/ $DATA_ROOT/static/
    ProxyPass /static/ !
    <Location /static/>
        Order deny,allow
        Allow from all
        Satisfy any
    </Location>
    # CAUTION: THE FOLLOWING LINES ALLOW PUBLIC ACCESS TO ANY UPLOADED CONTENT
    Alias /media/ $DATA_ROOT/media/
    # the right value is provided by "moneta-ctl config python | grep MEDIA_ROOT"
    ProxyPass /media/ !
    <Location /media/>
        Order deny,allow
        Allow from all
        Satisfy any
    </Location>
    ProxyPass / http://localhost:8131/
    ProxyPassReverse / http://localhost:8131/
    DocumentRoot $DATA_ROOT/static/
    # the right value is provided by "moneta-ctl config python | grep STATIC_ROOT"
    ServerSignature off
    # the optional two following lines are useful
    # for keeping uploaded content  private with good performance
    XSendFile on
    XSendFilePath $DATA_ROOT/media/
    # the right value is provided by "moneta-ctl config python | grep MEDIA_ROOT"
    # in older versions of XSendFile (<= 0.9), use XSendFileAllowAbove On
</VirtualHost>
EOF
sudo mkdir $DATA_ROOT
sudo chown -R www-data:www-data $DATA_ROOT
sudo a2ensite moneta.conf
sudo apachectl -t
sudo apachectl restart
```

If you want HTTP authentication, be sure to ensure that */core/p/* and */repo/p/* are publicly available. These URLs are used by packaging tools that do not use such authentication.

If you want to use SSL:

```
sudo apt-get install apache2 libapache2-mod-xsendfile
PEM=/etc/apache2/`hostname -f`.pem
# ok, I assume that you already have your certificate
sudo a2enmod headers proxy proxy_http ssl
openssl x509 -text -noout < $PEM
sudo chown www-data $PEM
sudo chmod 0400 $PEM

sudo apt-get install libapache2-mod-auth-kerb
KEYTAB=/etc/apache2/http.`hostname -f`.keytab
# ok, I assume that you already have your keytab
sudo a2enmod auth_kerb
cat << EOF | sudo ktutil
rkt $KEYTAB
list
```

(continues on next page)

(continued from previous page)

```
quit
EOF
sudo chown www-data $KEYTAB
sudo chmod 0400 $KEYTAB

SERVICE_NAME=moneta.example.org
cat << EOF | sudo tee /etc/apache2/sites-available/moneta.conf
<VirtualHost *:80>
    ServerName $SERVICE_NAME
    RedirectPermanent / https://$SERVICE_NAME/
</VirtualHost>
<VirtualHost *:443>
    ServerName $SERVICE_NAME
    SSLCertificateFile $PEM
    SSLEngine on
    Alias /static/ $DATA_ROOT/static/
    ProxyPass /static/ !
    <Location /static/>
        Order deny,allow
        Allow from all
        Satisfy any
    </Location>
    # CAUTION: THE FOLLOWING LINES ALLOW PUBLIC ACCESS TO ANY UPLOADED CONTENT
    Alias /media/ $DATA_ROOT/media/
    # the right value is provided by "moneta-ctl config python | grep MEDIA_ROOT"
    ProxyPass /media/ !
    <Location /media/>
        Order deny,allow
        Allow from all
        Satisfy any
    </Location>
    ProxyPass / http://localhost:8131/
    ProxyPassReverse / http://localhost:8131/
    DocumentRoot $DATA_ROOT/static/
    # the right value is provided by "moneta-ctl config python | grep STATIC_ROOT"
    ServerSignature off
    RequestHeader set X_FORWARDED_PROTO https
    <Location />
        AuthType Kerberos
        AuthName "Moneta"
        KrbAuthRealms EXAMPLE.ORG example.org
        Krb5Keytab $KEYTAB
        KrbLocalUserMapping On
        KrbServiceName HTTP
        KrbMethodK5Passwd Off
        KrbMethodNegotiate On
        KrbSaveCredentials On
        Require valid-user
        RequestHeader set REMOTE_USER %{REMOTE_USER}s
    </Location>
    # the optional two following lines are useful
    # for private uploaded content and good performance
    XSendFile on
    XSendFilePath $DATA_ROOT/media/
    # the right value is provided by "moneta-ctl config python | grep MEDIA_ROOT"
    # in older versions of XSendFile (<= 0.9), use XSendFileAllowAbove On
    <Location /core/p/>
```

(continues on next page)

(continued from previous page)

```
        Order deny,allow
        Allow from all
        Satisfy any
    </Location>
    <Location /repo/p/>
        Order deny,allow
        Allow from all
        Satisfy any
    </Location>
</VirtualHost>
EOF
sudo mkdir $DATA_ROOT
sudo chown -R www-data:www-data $DATA_ROOT
sudo a2ensite moneta.conf
sudo apachectl -t
sudo apachectl restart
```

1.3.6 Application

Now, it's time to install Moneta:

```
pip install setuptools --upgrade
pip install pip --upgrade
pip install moneta psycopg2
mkdir -p $VIRTUAL_ENV/etc/moneta
cat << EOF > $VIRTUAL_ENV/etc/moneta/settings.ini
[global]
data = $HOME/moneta
[database]
db = moneta
engine = postgresql
host = localhost
password = 5trongp4ssw0rd
port = 5432
user = moneta
EOF
chmod 0400 $VIRTUAL_ENV/etc/moneta/settings.ini
# protect passwords in the config files from by being readable by everyone
moneta-ctl collectstatic --noinput
moneta-ctl migrate
moneta-ctl createsuperuser
```

On VirtualBox, you may need to install rng-tools to generate enough entropy for GPG keys:

```
sudo apt-get install rng-tools
echo "HRNGDEVICE=/dev/urandom" | sudo tee -a /etc/default/rng-tools
sudo service rng-tools restart
```

1.3.7 supervisor

Supervisor can be used to automatically launch moneta:

```
sudo apt-get install supervisor
cat << EOF | sudo tee /etc/supervisor/conf.d/moneta.conf
[program:moneta_aiohttp]
command = $VIRTUAL_ENV/bin/moneta-ctl server
user = moneta
EOF
sudo service supervisor stop
sudo service supervisor start
```

Now, Supervisor should start moneta after a reboot.

1.3.8 systemd

You can also use systemd in most modern Linux distributions to launch moneta:

```
cat << EOF | sudo tee /etc/systemd/system/moneta-web.service
[Unit]
Description=Moneta web process
After=network.target

[Service]
User=moneta
Group=moneta
WorkingDirectory=$DATA_ROOT/
ExecStart=$VIRTUAL_ENV/bin/moneta-ctl server
ExecReload=/bin/kill -s HUP \$MAINPID
ExecStop=/bin/kill -s TERM \$MAINPID
Restart=on-failure

[Install]
WantedBy=multi-user.target
EOF
systemctl enable moneta-web.service
sudo service moneta-web
```

1.4 Python hosting

1.4.1 Heroku

First, you need to prepare your Heroku deployment:

```
mkdir heroku-hosting
cd heroku-hosting
pip install pipenv
heroku login
```

Now, a few files are required:

```
# create the Pipfile for downloading and installing Moneta
cat << EOF > Pipfile
[[source]]
url = "https://pypi.python.org/simple"
verify_ssl = true
```

(continues on next page)

(continued from previous page)

```
[packages]
moneta = '*'
django-redis-sessions = "*"
django-redis = "*"
psycopg2 = "*"
[requires]
python_version = "3.6"
EOF

# create a simple manage.py for the automatic collectstatic command
cat << EOF > manage.py
#!/usr/bin/env python
from djangofloor.scripts import django, set_env

set_env(command_name='moneta-ctl')
django()
EOF

cat << EOF > local_settings.ini
[global]
data = ./
EOF

# create the Procfile with required processes
cat << EOF > Procfile
web: moneta-ctl server
EOF
```

You can now deploy using git:

```
git init
git add .
git commit -m 'initial commit'
heroku create
# if you prefer SSH, you should use the following line
heroku create --ssh-git
# the metadata feature is required to provide the Heroku app name in the env
heroku labs:enable runtime-dyno-metadata
git push heroku master
```

You can add a Redis app to use efficient session and cache storages..

```
heroku addons:create heroku-redis -a heroku_app_name
```

If you do not use Redis for sessions and cache, you must remove *django-redis-sessions* and *django-redis* from your Pipfile.

Once deployed, you can prepare the database, create an administrator or open Python shell:

```
heroku run moneta-ctl migrate
heroku run moneta-ctl createsuperuser
heroku run moneta-ctl shell
```

Finally, you need to create at least one worker:

```
heroku ps:scale web=1
```

1.4.2 Gandi

Moneta must be locally installed (in a virtualenv) to prepare the deployment on a Gandi host.

```
mkdir gandi-hosting
pip install moneta
cd gandi-hosting
cat << EOF > gandi.yml
python:
  version: 3.6
EOF
cat << EOF > wsgi.py
import os
from djangofloor.scripts import get_application

os.environ['LC_ALL']="en_US.UTF-8"
os.environ['LC_LANG']="en_US.UTF-8"
application = get_application(command_name='moneta-ctl')
EOF
cat << EOF > requirements.txt
moneta
EOF
cat << EOF > local_settings.ini
[global]
data = ./
server_url = https://www.example.com/
EOF
moneta-ctl collectstatic --noinput
git add .
git commit -am 'initial commit'
```

1.5 Complete configuration

1.5.1 Configuration options

You can look current settings with the following command:

```
moneta-ctl config ini -v 2
```

You can also display the actual list of Python settings (for more complex tweaks):

```
moneta-ctl config python -v 2
```

Here is the complete list of settings:

```
[auth]
allow_basic_auth = true
      # Set to "true" if you want to allow HTTP basic auth, using the Django database.
create_users = false
      # Set to "false" if users cannot create their account themselvers, or only if␣
↪existing users can by authenticated by the reverse-proxy.
ldap_bind_dn = cn=admin,dc=example,dc=com
      # Bind dn for LDAP authentication
ldap_bind_password = toto
      # Bind password for LDAP authentication
```

(continues on next page)

(continued from previous page)

```
ldap_deny_group =
      # authentication is denied for users belonging to this group. Must be something␣
↪like "cn=disabled,ou=groups,dc=example,dc=com".
ldap_direct_bind = uid=%(user)s,ou=People,dc=example,dc=com
      # Set it for a direct LDAP bind and to skip the LDAP search, like "uid=%
↪%(user)s,ou=users,dc=example,dc=com". %%(user)s is the only allowed variable and␣
↪the double "%%" is required in .ini files.
ldap_email_attribute =
      # LDAP attribute for the user's email, like "email".
ldap_filter = (uid=%(user)s)
      # Filter for LDAP authentication, like "(uid=%%(user)s)" (the default), the␣
↪double "%%" is required in .ini files.
ldap_first_name_attribute = givenName
      # LDAP attribute for the user's first name, like "givenName".
ldap_group_search_base = ou=Groups,dc=example,dc=com
      # Search base for LDAP groups, like "ou=groups,dc=example,dc=com"
ldap_group_type = posix
      # Type of LDAP groups. Valid choices: "posix", "nis", "GroupOfNames",
↪"NestedGroupOfNames", "GroupOfUniqueNames", "NestedGroupOfUniqueNames",
↪"ActiveDirectory", "NestedActiveDirectory", "OrganizationalRole",
↪"NestedOrganizationalRole"
ldap_is_active_group = cn=active,ou=Groups,dc=example,dc=com
      # LDAP group DN for active users, like "cn=active,ou=groups,dc=example,dc=com"
ldap_is_staff_group = cn=staff,ou=Groups,dc=example,dc=com
      # LDAP group DN for staff users, like "cn=staff,ou=groups,dc=example,dc=com".
ldap_is_superuser_group = cn=superusers,ou=Groups,dc=example,dc=com
      # LDAP group DN for superusers, like "cn=superuser,ou=groups,dc=example,dc=com".
ldap_last_name_attribute = sn
      # LDAP attribute for the user's last name, like "sn".
ldap_mirror_groups = true
      # Mirror LDAP groups at each user login
ldap_require_group = cn=active,ou=Groups,dc=example,dc=com
      # only authenticates users belonging to this group. Must be something like
↪"cn=enabled,ou=groups,dc=example,dc=com".
ldap_server_url = ldap://localhost:12389/
      # URL of your LDAP server, like "ldap://ldap.example.com". Python packages
↪"pyldap" and "django-auth-ldap" must be installed.Can be used for retrieving␣
↪attributes of users authenticated by the reverse proxy
ldap_start_tls = false
      # Set to "true" if you want to use StartTLS.
ldap_user_search_base = ou=People,dc=example,dc=com
      # Search base for LDAP authentication by direct after an search, like "ou=users,
↪dc=example,dc=com".
local_users = true
      # Set to "false" to deactivate local database of users.
pam = false
      # Set to "true" if you want to activate PAM authentication
radius_port =
      # port of the Radius server.
radius_secret =
      # Shared secret if the Radius server
radius_server =
      # IP or FQDN of the Radius server. Python package "django-radius" is required.
remote_user_groups = Users
      # Comma-separated list of groups, for new users that are automatically created␣
↪when authenticated by remote_user_header. Ignored if groups are read from a LDAP␣
↪server.
```

(continues on next page)

(continued from previous page)

```
remote_user_header = HTTP_REMOTE_USER
      # Set it if the reverse-proxy authenticates users, a common value is "HTTP_
↪REMOTE_USER". Note: the HTTP_ prefix is automatically added, just set REMOTE_USER␣
↪in the reverse-proxy configuration.
session_duration = 1209600
      # Duration of the connection sessions (in seconds, default to 1,209,600 s / 14␣
↪days)
social_providers = github
      # Comma-separated OAuth2 providers, among "baidu","angellist","openid","douban",
↪"persona","bitbucket","hubic","dropbox","daum","stackexchange","untappd","pinterest
↪","windowslive","google","dropbox_oauth2","fivehundredpx","vk","amazon","line",
↪"robinhood","vimeo","eveonline","discord","slack","twentythreeandme","edmodo","asana
↪","soundcloud","weixin","digitalocean","fxa","linkedin","twitter","linkedin_oauth2",
↪"auth0","odnoklassniki","bitbucket_oauth2","feedly","basecamp","shopify","mailru",
↪"weibo","draugiem","naver","mailchimp","paypal","facebook","gitlab","reddit",
↪"spotify","stripe","xing","twitch","orcid","github","kakao","bitly","flickr",
↪"coinbase","evernote","tumblr","foursquare","instagram". "django-allauth" package␣
↪must be installed.

[cache]
db = 2
      # Database number (redis only).
      # Python package "django-redis" is also required to use Redis.
engine = redis
      # cache storage engine ("locmem", "redis" or "memcache") Valid choices: "redis",
↪ "memcache", "locmem", "file"
host = localhost
      # cache server host (redis or memcache)
password =
      # cache server password (if required by redis)
port = 6379
      # cache server port (redis or memcache)

[database]
db = moneta
      # Main database name (or path of the sqlite3 database)
engine = postgresql
      # Main database engine ("mysql", "postgresql", "sqlite3", "oracle", or the␣
↪dotted name of the Django backend)
host = localhost
      # Main database host
password = 5trongp4ssw0rd
      # Main database password
port = 5432
      # Main database port
user = moneta
      # Main database user

[email]
from = system@19pouces.net
      # Displayed sender email
host = auth.smtp.1and1.fr
      # SMTP server
password = ao2-P_FtETUDcRta
      # SMTP password
port = 587
      # SMTP port (often 25, 465 or 587)
```

(continues on next page)

(continued from previous page)

```
use_ssl = false
      # "true" if your SMTP uses SSL (often on port 465)
use_tls = true
      # "true" if your SMTP uses STARTTLS (often on port 587)
user = system@19pouces.net
      # SMTP user

[global]
admin_email = admin@moneta.example.org
      # e-mail address for receiving logged errors
data = $DATA_ROOT
      # where all data will be stored (static/uploaded/temporary files, ...). If you␣
↪change it, you must run the collectstatic and migrate commands again.
language_code = fr-fr
      # default to fr_FR
listen_address = localhost:8131
      # address used by your web server.
log_directory = $DATA_ROOT/log/
      # Write all local logs to this directory.
log_remote_access = true
      # If true, log of HTTP connections are also sent to syslog/logd
log_remote_url =
      # Send logs to a syslog or systemd log daemon.
      # Examples: syslog+tcp://localhost:514/user, syslog:///local7, syslog:///dev/
↪log/daemon, logd:///project_name
server_url = http://moneta.example.org
      # Public URL of your website.
      # Default to "http://{listen_address}/" but should be different if you use a␣
↪reverse proxy like Apache or Nginx. Example: http://www.example.org/.
ssl_certfile =
      # Public SSL certificate (if you do not use a reverse proxy with SSL)
ssl_keyfile =
      # Private SSL key (if you do not use a reverse proxy with SSL)
time_zone = Europe/Paris
      # default to Europe/Paris
use_apache = true
      # "true" if Apache is used as reverse-proxy with mod_xsendfile.The X-SENDFILE␣
↪header must be allowed from file directories
use_nginx = False
      # "true" is nginx is used as reverse-proxy with x-accel-redirect.The media␣
↪directory (and url) must be allowed in the Nginx configuration.

[gnupg]
home = $DATA_ROOT/gpg/
      # Path of the GnuPG secret data
keyid =
      # ID of the GnuPG key
path = /usr/local/bin/gpg
      # Path of the gpg binary

[server]
graceful_timeout = 25
      # After receiving a restart signal, workers have this much time to finish␣
↪serving requests. Workers still alive after the timeout (starting from the receipt␣
↪of the restart signal) are force killed.
keepalive = 5
      # After receiving a restart signal, workers have this much time to finish␣
↪serving requests. Workers still alive after the timeout (starting from the receipt␣
↪of the restart signal) are force killed.
```

(continues on next page)

(continued from previous page)

```
max_requests = 10000
      # The maximum number of requests a worker will process before restarting.
processes = 2
      # The number of web server processes for handling requests.
threads = 2
      # The number of web server threads for handling requests.
timeout = 35
      # Web workers silent for more than this many seconds are killed and restarted.

[sessions]
db = 1
      # Database number of the Redis sessions DB
      # Python package "django-redis-sessions" is required.
host = localhost
      # Redis sessions DB host
password =
      # Redis sessions DB password (if required)
port = 6379
      # Redis sessions DB port
```

If you need more complex settings, you can override default values (given in *djangofloor.defaults* and *moneta.defaults*) by creating a file named */moneta/settings.py*.

1.5.2 Optional components

Efficient page caching

You just need to install *django-redis*. Settings are automatically changed for using a local Redis server (of course, you can change it in your config file).

```
pip install django-redis
```

Faster session storage

You just need to install *django-redis-sessions* for storing sessions into user sessions in Redis instead of storing them in the main database. Redis is not designed to be backuped; if you loose your Redis server, sessions are lost and all users must login again. However, Redis is faster than your main database server and sessions take a huge place if they are not regularly cleaned. Settings are automatically changed for using a local Redis server (of course, you can change it in your config file).

```
pip install django-redis-sessions
```

1.5.3 Debugging

If something does not work as expected, you can look at logs (check the global configuration for determining their folder) or try to run the server interactively:

```
sudo service supervisor stop
sudo -H -u moneta -i
workon moneta
moneta-ctl check
```

(continues on next page)

(continued from previous page)

```
moneta-ctl config ini
moneta-ctl server
```

You can also enable the DEBUG mode which is more verbose (and displays logs to stdout):

```
FILENAME=`easydemo-ctl config ini -v 2 | grep -m 1 ' - .ini file' | cut -d '"' -f 2 |␣
↪sed  's/.ini$/.py/'`
echo "DEBUG = True" >> $FILENAME
moneta-ctl runserver
```

1.6 Changes

1.7 Changelog history

1.7.1 1.8.7 (2017/11/26)

- new documentation
- search