DjNRO

Release 0.9

July 14, 2015

Contents

DjNRO = Django + NRO

CHAPTER 1

About

In the eduroam world, NRO stands for National Roaming Operator. Maintaining and managing a local eduroam database is quite an important responsibility of an eduroam NRO. eduroam.org periodically polls and gathers information from all participating domains. Information is provided upstream, in a structured way (XML format) and consists of participating institutions' data, location data along with monitoring data - though provisioning of monitoring data has been superseeded by the f-Ticks mechanism.

The source of information should be the local eduroam database. So, changes to the database should be reflected to the XML files. New eduroam locations, changes in contacts and information about each location should be up-to-date so as to ease the eduroam usage and assist eduroam users whenever they need support.

DjNRO is a Django platform that eases the management process of a National Roaming Operator. DjNRO complies with the eduroam database and the eduroam XSDs. Thus, apart from domain management, it can generate the necessary xml files for eduroam.org monitoring.

DjNRO is more than keeping eduroam.org updated with data.

In essence it is a distributed management application. It is distributed in the sense that information about each institution locations and services is kept up-to-date by each local eduroam administrator. Keeping in pace with eduroam's federated nature, our implementation uses federated authentication/authorisation mechanisms, namely Shibboleth. In case Shibboleth is not an option for an institution, a social media auth mechanism comes in handy. The local institution eduroam administrators can become DjNRO admins. Local eduroam administrators register to the platform via Shibboleth or social media auth. The NRO's responsibility is to activate their accounts.

From then on they can manage their eduroam locations, contact points and institution information. The administrative interface especially the locations management part, is heavily implemented with Google Maps. This makes editing easier, faster and accurate.

Installation and customization is fairly easy and is described in the following sections.

Attention: Installation instructions assume a clean Debian Wheezy with Django 1.4

Currently the source code is availiable at code.grnet.gr and can be cloned via git:

```
git clone https://code.grnet.gr/git/djnro
```

The Greek eduroam webpage is a living example of DjNRO: eduroamlgr

CHAPTER 2

Features

- Allow your local eduroam admins to edit their local eduroam data (AP locations, server params, etc)
- Visualize the information via Google Maps
- Eduroam world maps overview via daily update on eduroam.org KML file
- Find your closest eduroam in the world
- **New** Allow for eduroam CAT institution enrollments
- **New** Extract contact info for mailing list creation
- **New** Server monitoring data

Bootstrap CSS framework with responsive design makes it work on every device

CHAPTER 3

Requirements

3.1 Required Packages

3.1.1 Dependencies

DjNRO heavily depends on the following:

- Python (<3 & >=2.6)
- Django (>=1.2) - python-django
- memcached
- python-django-extensions
- python-mysqldb (If you wish to use MySQL as the DB backend)
- mysql-client
- python-ipaddr
- python-django-south (For database migrations). If you deploy MySQL >=5.5 and earlier versions of south (< 0.7.5), you are advised to upgrade to South >=0.7.5, as you may suffer from this bug
- python-django-tinymce (Flatpages editing made easier)
- python-memcache (Yeap! You need that for Google maps locations caching)
- python-django-registration (User activation made easy)
- apache2 (We suggest apache with mod_rewrite enabled - use your preferred server)
- libapache2-mod-wsgi
- libapache2-mod-shib2 : The server should be setup as a Shibboleth SP
- A mail server - Tested with exim

3.1.2 Conditional Dependencies

- gettext: only if one will be editing and compiling translations
- python-django-auth-ldap: if ldap authentication backend will be used.

Django Social Auth

User authentication via social media is carried out by the python-django-social-auth python-django-social-auth package. We have included python-django-social-auth 0.7.18 in repository because DjNRO requires WrongBackend from social_auth.exceptions; this does not exist in 0.7.0 which ships with Debian Wheezy.

3.1.3 Django Social Auth: Requirements - Dependencies

- OpenId support depends on python-openid
- OAuth support depends on python-oauth2

CHAPTER 4

Installation

4.1 Installation/Configuration

Contents

Note: Installation instructions assume a clean Debian Wheezy with Django 1.4

Assuming that you have installed all the requirements described in *Required Packages* you can install the DjNRO project.

The software is published at code.grnet.gr and can be downloaded using git:

```
git clone https://code.grnet.gr/git/djnro
```

It is also available on GitHub:

```
https://github.com/grnet/djnro/
```

4.1.1 Project & Local Settings

Attention: In version 0.9 settings were split in two parts: settings.py and local_settings.py

The file settings.py contains settings distributed by the project, which should normally not be necessary to modify. Options specific to the particular installation must be configured in local_settings.py. This file must be created by copying local_settings.py.dist:

```
cd djnro
cp djnro/local_settings.py.dist djnro/local_settings.py
```

The following variables/settings need to be altered or set:

Set Admin contacts:

```
ADMINS = (
     ('Admin', 'admin@example.com'),
)
```

Set the database connection params:

```
DATABASES = {
    ...
}
```

For a production instance and once DEBUG is set to False set the ALLOWED_HOSTS:

```
ALLOWED_HOSTS = ['.example.com']
```

Set your timezone and Languages:

```
TIME_ZONE = 'Europe/Athens'

LANGUAGES = (
    ('el', _('Greek')),
    ('en', _('English')),
)
```

Set your static root and url:

```
STATIC_ROOT = '/path/to/static'
STATIC_URL = 'http://www.example.com/static'
```

Attention: The STATIC_URL setting works only if DEBUG=False. For more see the Django docs.

Set the secret key:

```
SECRET_KEY = '<put something really random here, eg. %$#%@#$^2312351345#$%3452345@#$%@#$234#@$hhzdavi
```

Django social auth needs changes in the Authentication Backends depending on which social auth you want to enable:

```
AUTHENTICATION_BACKENDS = (
    'djnro.djangobackends.shibauthBackend.shibauthBackend',
        ...
        'django.contrib.auth.backends.ModelBackend',
)
```

Set your template dirs:

```
TEMPLATE_DIRS = (
    "/example/templates",
)
```

As the application includes a “Nearest eduroam” functionality, global eduroam service locations are harvested from the KML file published at eduroam.org:

```
EDUROAM_KML_URL = 'http://monitor.eduroam.org/kml/all.kml'
```

Depending on your AAI policy set an appropriate authEntitlement:

```
SHIB_AUTH_ENTITLEMENT = 'urn:mace:example.com:pki:user'
```

Mail server parameters:

```
SERVER_EMAIL = "Example domain eduroam Service <noreply@example.com>"
EMAIL_SUBJECT_PREFIX = "[eduroam] "
```

NRO contact mails:

```
NOTIFY_ADMIN_MAILS = ["mail1@example.com", "mail2@example.com"]
```

Set your cache backend (if you want to use one). For production instances you can go with memcached. For development you can keep the provided dummy instance:

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
```

Models Name_i18n and URL_i18n include a language choice field If languages are the same with LANGUAGES variable, simply do URL_NAME_LANGS = LANGUAGES else set your own:

```
URL_NAME_LANGS = (
        ('en', 'English' ),
        ('el', 'Ελληνικ'),
    )
```

NRO specific parameters. These affect HTML templates:

```
# Frontend country specific vars, eg. Greece
NRO_COUNTRY_NAME = _('My Country')
# Variable used by context_processor to display the "eduroam | <country_code>" in base.html
NRO_COUNTRY_CODE = 'gr'
# main domain url used in right top icon, eg. http://www.grnet.gr
NRO_DOMAIN_MAIN_URL = "http://www.example.com"
# provider info for footer
NRO_PROV_BY_DICT = {"name": "EXAMPLE DEV TEAM", "url": "http://devteam.example.com"}
#NRO social media contact (Use: // to preserve https)
NRO_PROV_SOCIAL_MEDIA_CONTACT = [
                                {"url":"//soc.media.url", "icon":"icon.png", "name":"NAME1(eg. Faceb
                                {"url":"//soc.media.url", "icon":"icon.png",  "name":"NAME2(eg. Twitt
                                ]
# map center (lat, lng)
MAP_CENTER = (36.97, 23.71)
#Helpdesk, used in base.html:
NRO_DOMAIN_HELPDESK_DICT = {"name": _("Domain Helpdesk"), 'email':'helpdesk@example.com', 'phone': '1
```

Set the Realm country for REALM model:

```
#Countries for Realm model:
REALM_COUNTRIES = (
             ('country_2letters', 'Country' ),
            )
```

Attribute map to match your AAI policy and SSO software (typically Shibboleth SP):

```
#Shibboleth attribute map
SHIB_USERNAME = ['HTTP_EPPN']
SHIB_MAIL = ['mail', 'HTTP_MAIL', 'HTTP_SHIB_INETORGPERSON_MAIL']
SHIB_FIRSTNAME = ['HTTP_SHIB_INETORGPERSON_GIVENNAME']
SHIB_LASTNAME = ['HTTP_SHIB_PERSON_SURNAME']
SHIB_ENTITLEMENT = ['HTTP_SHIB_EP_ENTITLEMENT']
```

Django Social Auth parameters:

```
TWITTER_CONSUMER_KEY = ''
TWITTER_CONSUMER_SECRET = ''

FACEBOOK_APP_ID = ''
FACEBOOK_API_SECRET = ''

LINKEDIN_CONSUMER_KEY        = ''
LINKEDIN_CONSUMER_SECRET     = ''

YAHOO_CONSUMER_KEY = ''
YAHOO_CONSUMER_SECRET = ''

GOOGLE_SREG_EXTRA_DATA = []
```

New in version 0.9.

DjNRO provides limited integration with eduroam CAT (Configuration Assistant Tool). Institution administrators can automatically provision their institution to CAT without the intervention of the federation (NRO) administrator.

In order to enable this functionality, you must list at least one instance and the corresponding description in CAT_INSTANCES. Beware that pages accessible by end users currently only show CAT information for the instance named *production*.

You must also set the following parameters for each CAT instance in CAT_AUTH:

- CAT_API_KEY: API key for authentication to CAT
- CAT_API_URL: API endpoint URL
- CAT_PROFILES_URL: Base URL for Intitution Download Area pages
- CAT_FEDMGMT_URL: URL For Federation Overview page (currently not in use)

```
CAT_INSTANCES = (
    ('production', 'cat.eduroam.org'),
    ('testing', 'cat-test.eduroam.org'),
)

CAT_AUTH = {
    'production': {
        "CAT_API_KEY": "<provided API key>",
        "CAT_API_URL": "https://cat.eduroam.org/admin/API.php",
        "CAT_PROFILES_URL": "https://cat.eduroam.org/",
        "CAT_FEDMGMT_URL": "https://cat.eduroam.org/admin/overview_federation.php"
    },
    'testing': {
        "CAT_API_KEY": "<provided API key>",
        "CAT_API_URL": "https://cat-test.eduroam.org/test/admin/API.php",
        "CAT_PROFILES_URL": "https://cat-test.eduroam.org/test",
        "CAT_FEDMGMT_URL": "https://cat-test.eduroam.org/test/admin/overview_federation.php"
    },
}
```

For more information about eduroam CAT, you may read: A guide to eduroam CAT for federation administrators.

In case one wants to extend some of the settings only for the local instance, they can prepend 'EXTRA_' on the attribute they want to extend. For example:

```
EXTRA_INSTALLED_APPS = (
        'django_debug_toolbar',
)
```

4.1.2 Database Sync

Once you are done with local_settings.py run:

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

Create a superuser, it comes in handy. And then run south migration to complete:

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

Now you should have a clean database with all the tables created.

4.1.3 Running the server

We suggest using Apache and mod_wsgi. Below is an example configuration:

```
# Tune wsgi daemon as necessary: processes=x threads=y
WSGIDaemonProcess djnro display-name=%{GROUP} python-path=/path/to/djnro/

<VirtualHost *:443>
        ServerName              example.com

        Alias           /static /path/to/djnro/static
        WSGIScriptAlias /       /path/to/djnro/djnro/wsgi.py
        <Directory /path/to/djnro/djnro>
                <Files wsgi.py>
                    WSGIProcessGroup djnro
                    Order deny,allow
                    Allow from all
                </Files>
        </Directory>

        SSLEngine on
        SSLCertificateFile      ...
        SSLCertificateChainFile ...
        SSLCertificateKeyFile   ...

        # Shibboleth SP configuration
        ShibConfig      /etc/shibboleth/shibboleth2.xml
        Alias           /shibboleth-sp  /usr/share/shibboleth

        # SSO through Shibboleth SP:
        <Location /login>
                AuthType shibboleth
                ShibRequireSession On
                ShibUseHeaders On
                require valid-user
        </Location>
```

```
        <Location /Shibboleth.sso>
                SetHandler shib
        </Location>
</VirtualHost>
```

Info: It is strongly recommended to allow access to `/(admin|overview|alt-login)` *ONLY* from trusted subnets.

Once you are done, restart apache.

4.1.4 Fetch KML

A Django management command, named fetch_kml, fetches the KML document and updates the cache with eduroam service locations. It is suggested to periodically run this command in a cron job in order to keep the map up to date:

```
./manage.py fetch_kml
```

4.1.5 Initial Data

In order to start using DjNRO you need to create a Realm record for your NRO along with one or more contacts linked to it. You can visit the Django admin interface (`https://<hostname>/admin`) and add a Realm (remember to set REALM_COUNTRIES in local_settings.py). In DjNRO the NRO sets the environment for the institution eduroam admins. Therefore the NRO has to insert the initial data for his/her clients/institutions in the *Institutions* Model, again using the Django admin interface. As an alternative, you can copy your existing `institution.xml` to `/path/to/djnro` and run the following to import institution data:

```
./manage.py parse_instituion_xml
```

4.1.6 Exporting Data

DjNRO can export data in formats suitable for use by other software.

XML documents conforming to the eduroam database schemata are exported at the following URLs, as required for harvesting by eduroam.org:

```
/general/realm.xml
/general/institution.xml
/usage/realm_data.xml
```

New in version 0.9.

A list of institution administrators can be exported in CSV format or a plain format suitable for use by a mailing list (namely Sympa). This data is available through:

- a management comand (`./manage.py contacts`), which defaults to CSV output (currently with headers in Greek!) and can switch to plain output using `--mail-list`.
- a view (`adminlist`), which only supports output in the latter plain text format.

Likewise, data that can be used as input for automatic configuration of *Federation Level RADIUS Servers (FLRS)* can be exported in YAML/JSON format, through:

- a management command (`./manage.py servdata`)
- a view (`sevdata`)

Output format defaults to YAML and can be overriden respectively:

- by using `--output=json`
- by sending an `Accept: application/json` HTTP header

We also provide a sample script for reading this data (`extras/servdata_consumer.py`) along with templates (in the same directory) for producing configuration suitable for FreeRADIUS and radsecproxy. This script requires the following python packages:

- python-requests
- python-yaml
- python-mako (for the templates)

Take the time to read the default settings at the top of the script and run it with `--help`. The templates are based on assumptions that may not match your setup; they are mostly provided as a proof of concept.

Attention: The `adminlist` and `servdata` views are commented out by default in `djnro/urls.py`. Make sure you protect them (SSL, ACL and/or authentication) at the HTTP server before you enable them, as they may expose private/sensitive data.

4.1.7 Next Steps (Set your Logo)

The majority of branding is done via the NRO variables in local_settings.py. You might also want to change the logo of the application. Within the static/img/eduroam_branding folder you will find the XCF files logo_holder, logo_small. Edit with Gimp according to your needs and export to logo_holder.png and logo_small.png at the same path. To change the domain logo on top right, replace the static/img/right_logo_small.png file with your own logo (86x40).

4.1.8 Upgrade Instructions

- Backup your `settings.py` file and any local modifications.
- Update the code.
- Copy `djnro/local_settings.py.dist` to `djnro/local_settings.py` and modify it to match your previous configuration.
- edit the apache configuration in order to work with the new location of wsgi and

set the python-path attribute.

- remove old wsgi file `/path/to/djnro/apache/django.wsgi` and parent directory
- remove django-extensions from *INSTALLED_APPS*
- Add timeout in cache configuration
- Make sure you have installed the following required packages (some of these introduced in 0.9):
  - python-oauth2
  - python-requests
  - python-lxml
  - python-yaml
- run `./manage.py migrate`

Attention: You had previously copied `urls.py.dist` to `urls.py`. This is no longer supported; we now use `djnro/urls.py`. URLs that provide sensitive data are disabled (commented out) by default. You may have to edit the file according to your needs.

4.1.9 Pip Support

We have added a requirements.txt file, tested for django 1.4.5. You can use it with `pip install -r requirements.txt`.

4.1.10 LDAP Authentication

If you want to use LDAP authentication, local_settings.py must be amended:

```
EXTRA_AUTHENTICATION_BACKENDS = (
        ...,
        'django_auth_ldap.backend.LDAPBackend',
        ...,
)

# LDAP CONFIG
import ldap
from django_auth_ldap.config import LDAPSearch, GroupOfNamesType
AUTH_LDAP_BIND_DN = ""
AUTH_LDAP_BIND_PASSWORD = ""
AUTH_LDAP_SERVER_URI = "ldap://foo.bar.org"
AUTH_LDAP_START_TLS = True
AUTH_LDAP_USER_SEARCH = LDAPSearch("ou=People, dc=bar, dc=foo",
ldap.SCOPE_SUBTREE, "(uid=%(user)s)")
AUTH_LDAP_USER_ATTR_MAP = {
      "first_name":"givenName",
      "last_name": "sn",
      "email": "mail
      }
# Set up the basic group parameters.
AUTH_LDAP_GROUP_SEARCH = LDAPSearch(
        "ou=Groups,dc=foo,dc=bar,dc=org",ldap.SCOPE_SUBTREE, objectClass=groupOfNames"
)
AUTH_LDAP_GROUP_TYPE = GroupOfNamesType()
AUTH_LDAP_USER_FLAGS_BY_GROUP = {
        "is_active": "cn=NOC, ou=Groups, dc=foo, dc=bar, dc=org",
        "is_staff": "cn=staff, ou=Groups, dc=foo, dc=bar, dc=org",
        "is_superuser": "cn=NOC, ou=Groups,dc=foo, dc=bar, dc=org"
}
```