Seeder Documentation

Release 0.1

Visgean Skeloru

Nov 06, 2018

Contents

This project is a web management tool for Czech web archive.

CHAPTER 1

Installation

Prerequisites:

- python 3.5
- python psycopg2 driver
- postgresql-devel
- mysql-devel
- libjpeg-devel
- zlib-devel
- python-devel
- gcc
- PIP
- virtualenv
- PostgreSQL
- nginx
- supervisor
- uwsgi

Virtualenv

Virtualenv is something like chroot for python libraries. Installation instructions: https://virtualenv.pypa.io/en/latest/installation.html . Then create virtualenv seeder: `$virtualenv seeder` You have to activate it every time before using python: `source seeder/bin/activate` .

Configuration

Firstly create *Seeder/settings/local_settings.py* according to template *Seeder/settings/local_settings.template.py*.

Then:

- set secret key, that should be something long and random

- set debugs to `False` for security reasons
- set allowed host variable - put there your domain name
- finally set the database username and password

Read more about settings at: https://docs.djangoproject.com/en/1.8/ref/settings/

nginx

After installing and configuring nginx create config file similar to *template.nginx.conf* in `/etc/nginx/sites-available/` and make a link to it in `/etc/nginx/sites-enabled`.

uwsgi

Put something like *template.uwsgi.conf* to `/etc/uwsgi/apps-available/` . and link to it from `/etc/uwsgi/apps-enabled/` .

supervisor

Put something like *template.supervisor.conf* to `/etc/supervisor/conf.d/` .

Cron

You need to run `python manage.py runcrons` periodically, this commands runs periodical tasks that takes care of various thins - screenshots, postponed voting rounds, expiring contracts...

So use something like this

```
0 * * * * source <virtualenv>/bin/activate && python <seeder>Seeder/manage.py␣
↪runcrons > <log_path>/django_cron.log
```

Manet

Install https://github.com/vbauer/manet with PhantomJS support. Note that it must be running in order to take screenshots. There are also some cases where manet fails horribly for no reason.

Final restart

After configuring all of the above lets restart servers

```
$ sudo service supervisor restart
$ sudo service nginx restart
```

The proceed with deploying.

CHAPTER 2

Docker Compose

For developing purposes you can use `docker-compose` which creates various dockers and networks them together. This setup in not secure and database might get deleted on accident.

Running up the containers

```
$ docker-compose up
```

this will run the runserver on localhost port 8000.

You will need to create your super user in order to log in:

```
$ docker-compose run web --rm ./manage.py createsuperuser
```

If you need to import data from legacy system put the raw sql file in `legacy_dumps` folder and run following command:

```
$ docker-compose run web --rm ./manage.py legacy_sync
```

If you add some new requirements you will need to rebuild the images with `docker-compose build` command. Even though the command for running the server will try to install latest requirements it won't affect other dockers so you will have trouble accessing any manage command.

Shortcut scripts

For easier development there are two scripts.

- `./drun migrate` will run `docker-compose run web --rm ./manage.py migrate`
- `crns` will run development server with service ports exposed - fixes `pdb` bugs

CHAPTER 3

Deploying

Deploying takes care of installing PIP packages, installing js packages and static files collecting.

Manual

You need to run following commands:

```
$ git pull
$ pip install -R requirements.txt -U
$ ./manage.py migrate
$ tx pull -a
$ ./manage.py compilemessages
$ ./manage.py collectstatic
```

Using Fabric

Local deploying can be executed server-side from seeder directory. To do this simply type (with active `seeder` virtualenv!)

```
seeder/Seeder $ fab deploy_locally
```

CHAPTER 4

Integration with legacy system

If you have filled out `legacy_database` in `settings.py` you can use

```
$ ./manage.py legacy_sync
```

This command will automatically run all the migrations. Note, not all data can be migrated, there are some broken relations in Contacts table.

Skipped tables:

- Correspondence
- CorrespondenceType
- Keywords
- KeywordsResources
- QaChecks
- QaChecksQaProblems
- QaProblems
- Roles
- Subcontracts

These tables were skipped because they did not have any meaningful representation in the project or they did not contain any data.

Links

For all the sources that have been imported you can find link to the legacy system so you can see things like comments that have not been migrated properly.

CHAPTER 5

Crons

Installation

Crontab can be installed via `manage.py` :

```
$ python3 manage.py crontab add
```

Contract expiry

This cron expires contracts that have a value in `valid_to` field. They also set source state to expired state - meaning that it wont be included in harvest. So it should be used wisely.

Voting round reviver

Revives voting rounds that have been postponed. It does not create new voting rounds, it only opens the old one.

Publisher communication cron

Cron that sends scheduled emails about contracts negotiation.

CHAPTER 6

Translation

Translation is happening here: https://www.transifex.com/projects/p/seeder/ Git ignores all translation files so you have to download translated strings every time. Simply run `pull_locales.sh` which will download and auto-compile translation strings.

To pull the messages you need to have create `Seeder/.transifexrc` file. Have a look at template `Seeder/.transifexrc_template`.

Updating

If you have updated the code and wish to translate new changes, run `push_locales.sh` and translate changes on transifex.

Languages

Define new languages in `settings/base.py` LANGUAGES variable.

CHAPTER 7

Terminology

If might get confusing sometimes to work with all those names. This all seems like some very odd farming project with terms like Harvests and seeds...

Sources

Sources are sort of publications - they can have multiple seeds = URLs, they have a publisher and they need to have assigned contract otherwise they might not be harvested.

Seeds

Seeds are just weird way how to say URL. Each seed has its own sources. Sources can have multiple seeds. Seeds have different rules how they can be harvested based on technical necessities.

Voting round

Process of deciding whether source should be archived or not. This process is repeated sometimes.

Curator

Somebody who checks the content of the archiving sources. Masters of the archive.

QA check

Quality assurance check that happens after source has been accepted to archive. This is a check mainly for the content changes and technical side of the harvesting.

Publishers

They publish sources. They need to sign a contract unless they have open source licence.

Harvests

Instance of an act of downloading seeds and archiving them. Might happen automatically in future.

Harvest blacklist

Some publishers don't want to be their resources harvested. So they are blacklisted. Miserable people those are.

Visibility blacklist

Some sites are harvested but they don't have contract yet so they must not ever be displayed on a web.

CHAPTER 8

Harvests

To schedule harvests from the start to the end of current year run

```
$ ./manage.py schedule_harvests
```

This will create harvests for all the frequencies that are being harvested. In future its possible that seeder will run harvests automatically.

CHAPTER 9

API

This document describes how to work with the Seeder APIs

Authentication

Authentication is based on token sent over the headers. You will need to get this token to do anything useful on production environment.

```
$ http POST :8000/api/token username=username password=heslo -vv
POST /api/token HTTP/1.1
Accept: application/json
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Length: 47
Content-Type: application/json
Host: localhost:8000
User-Agent: HTTPie/0.9.3

{
    "password": "heslo",
    "username": "username"
}

HTTP/1.0 200 OK
Allow: POST, OPTIONS
Content-Language: en
Content-Type: application/json
Date: Fri, 08 Apr 2016 00:14:39 GMT
Server: WSGIServer/0.1 Python/2.7.6
Vary: Accept-Language, Cookie
X-Frame-Options: SAMEORIGIN

{
    "token": "b4a3f506347adcdd51bc3c1e95449002384ab260"
}
```

Source endpoint

The most useful API is the source endpoint. This can be used to retrieve and update the source data.

The source url is on `/api/source/<id>`

[GET]

The get request will return document with following structure:


```
{
    "active": true,
    "aleph_id": "2121",
    "annotation": "document annotation",
    "category": 12,
    "comment": "internal comment",
    "created": "2016-02-06T00:41:45.453995Z",
    "frequency": 12,
    "id": 1,
    "issn": "1212-50125",
    "last_changed": "2016-04-07T22:45:41.873747Z",
    "mdt": "02",
    "name": "Source name",
    "publisher": {
        "active": true,
        "contacts": [
            {
                "active": true,
                "address": "Praha",
                "created": "2016-02-06T00:40:39.625087Z",
                "email": "redakce@example.com",
                "id": 1,
                "last_changed": "2016-02-06T00:40:39.625110Z",
                "name": "Petra",
                "phone": null,
                "position": null,
                "publisher": 1
            }
        ],
        "created": "2016-02-06T00:40:06.532276Z",
        "id": 1,
        "last_changed": "2016-02-06T00:40:06.532302Z",
        "name": "Example publisher"
    },
    "publisher_contact": 1,
    "screenshot": "http://localhost:8000/media/screenshots/1_04042016.png",
    "screenshot_date": "2016-04-04T00:37:20.388037Z",
    "seed": {
        "active": true,
        "budget": null,
        "calendars": false,
        "comment": "",
        "created": "2016-02-06T00:52:32.701084Z",
        "from_time": null,
        "gentle_fetch": "",
        "global_reject": false,
        "id": 322,
        "javascript": false,
        "last_changed": "2016-03-16T23:40:57.124311Z",
        "local_traps": false,
        "redirect": false,
```

```
        "robots": false,
        "state": "exc",
        "to_time": null,
        "url": "http://www.example.com",
        "youtube": false
    },
    "state": "success",
    "sub_category": 235,
    "suggested_by": null
}
```

For source and state values / meaning see `Seeder/source/constants.py` file.

[PATCH]

You can update the source document with the same structure as displayed in GET. You should only list the fields that you wish to update.

Following example shows partial update of the source document.

```
{
   "seed":{

      "url": "http://www.example.com",
      "global_reject": true
   },
   "name": "New source name",
   "sub_category": 231
}
```