Technical Documentation for the PYROS project (FGFT-CC)

HOWTO Format Redmine Wiki : http://www.redmine.org/projects/redmine/wiki/FrRedmineWikiFormatting

I - TODO

- installation sur windows
- gitlab
- séparation des BD Django et Pyros
- intégration dans Eclipse
- Intégration des modules Django déjà développés

I - DATABASE SCHEMA (v0.2.1)

{{thumbnail(PYROS_PDM_v021.png, size=300, title=Pyros data model)}}

II - Get the project (from gitlab)

Get the project from the terminal

git clone https://gitlab.irap.omp.eu/epallier/pyros.git PYROS

(or also : git clone git@gitlab.irap.omp.eu:epallier/pyros.git)

This creates a PYROS/ folder containing the project (with a .git/ subfolder for synchronization with the git repository)

If you just wanted a static copy of the project (without synchronization) just remove the .git/ folder:

$ rm -r .git/

You should obtain this structure:

```
PYROS/
├── REQUIREMENTS.txt
├── private/
│   └── venv_py35_pyros/
├── public/
│   └── static/
└── src/
    ├── manage.py
    ├── pyros/
    │   ├── __init__.py
    │   ├── __pycache__
    │   ├── settings.py
    │   ├── urls.py
    │   └── wsgi.py
    └── pyrosapp/
        ├── __init__.py
        ├── admin.py
        └── apps.py
```

```
    ├── migrations
    ├── models.py
    ├── tests.py
    └── views.py
```

Get the project from Eclipse

TODO:

III - INSTALLATION

Install MySql (only if necessary)

- Linux Ubuntu

```
$ sudo apt-get install mysql-server
$ sudo apt-get install mysql-client
```

- Linux CentOS

TODO:
```
$ sudo yum install mysql
...
```

- Mac OS X
  Install XAMPP
  (but you could also use the pre-installed Mac OS MySql)

TODO:

- Windows

Download and install the newest version on https://dev.mysql.com/downloads/installer/

Once installed, launch MySQL Installer. Clic on 'Add...' on the right.
In MySQLServers section, choose the newest, then clic on next.
Install and configure the server (just follow the installation guide).

Then launch mysql (via the Windows menu).

Install Python3.5 (only if necessary)

- Mac OS X :

1) Installer MacPort
(TODO: doc)

2) Installer le "port" python35
```
$ sudo port install python35
```

- Linux (Ubuntu) :

$ sudo add-apt-repository ppa:fkrull/deadsnakes
$ sudo apt-get update
$ sudo apt-get install python3.5

$ sudo pip install virtualenv

- Windows 7 :

Go to https://www.python.org/downloads/windows/ , choose the wanted version
On the wanted version's page, download Windows x86 executable installer

Run the executable
* On the first page, check "Add python3.5 to PATH"
* Choose "Install now" option

Open cmd (windows + R, cmd) :
$ python -m pip install --upgrade pip
$ pip install virtualenv

- Windows 10 :

TODO:

Create a Python3 virtual environment dedicated to the project (inside the project folder)

$ mkdir private/

$ cd private/

$ which python3.5 ("where python" for windows)
/opt/local/bin/python3.5

$ virtualenv-3.5 venv_py35_pyros -p /opt/local/bin/python3.5
=> creates a venv_py35_pyros/ folder inside PYROS/private/

Activate the python virtual environment (from inside the project)

$ pwd
.../PYROS/private

$ source ./venv_py35_pyros/bin/activate (venv_py35_pyros/Scripts/activate on Windows)

$ python -V

```
Python 3.5.1

$ which pip
.../PYROS/venv_py35_pyros/bin/pip

Upgrade pip to last version available:
$ pip install --upgrade pip
Collecting pip
  Downloading pip-8.1.1-py2.py3-none-any.whl (1.2MB)
Installing collected packages: pip
  Found existing installation: pip 7.1.2
    Uninstalling pip-7.1.2:
      Successfully uninstalled pip-7.1.2
Successfully installed pip-8.1.1

Upgrade wheel to last version available:
$ pip install --upgrade wheel
Collecting wheel
  Downloading wheel-0.29.0-py2.py3-none-any.whl (66kB)
Installing collected packages: wheel
  Found existing installation: wheel 0.24.0
    Uninstalling wheel-0.24.0:
      Successfully uninstalled wheel-0.24.0
Successfully installed wheel-0.29.0
```

Install the needed Python packages (from within the virtual environment)

First, be sure that the virtual environment is activated:

```
$ python -V
Python 3.5.1
```

- **Automatic Installation of all packages**

```
$ pip install -r REQUIREMENTS.txt
```

- **Or, manual installation of each package**
  - **Install Django** :

```
$ pip install django
Collecting django
  Downloading Django-1.9.4-py2.py3-none-any.whl (6.6MB)
Installing collected packages: django
Successfully installed django-1.9.4

$ pip install django-admin-tools
Collecting django-admin-tools
  Downloading django_admin_tools-0.7.2-py2.py3-none-any.whl (289kB)
Installing collected packages: django-admin-tools
Successfully installed django-admin-tools-0.7.2

$ pip install django-debug-toolbar
Collecting django-debug-toolbar
  Downloading django_debug_toolbar-1.4-py2.py3-none-any.whl (212kB)
Requirement already satisfied (use --upgrade to upgrade): Django>=1.7 in ./venv_py35_pyros/lib/python3.5/site-packages (from
django-debug-toolbar)
Collecting sqlparse (from django-debug-toolbar)
  Downloading sqlparse-0.1.19.tar.gz (58kB)
Building wheels for collected packages: sqlparse
  Running setup.py bdist_wheel for sqlparse ... done
  Stored in directory:
```

/Users/epallier/Library/Caches/pip/wheels/7b/d4/72/6011bb100dd5fc213164e4bbee13d4e03261dd54ce6a5de6b8
Successfully built sqlparse
Installing collected packages: sqlparse, django-debug-toolbar
Successfully installed django-debug-toolbar-1.4 sqlparse-0.1.19

$ pip install django-extensions
Collecting django-extensions
  Downloading django_extensions-1.6.1-py2.py3-none-any.whl (202kB)
Collecting six>=1.2 (from django-extensions)
  Downloading six-1.10.0-py2.py3-none-any.whl
Installing collected packages: six, django-extensions
Successfully installed django-extensions-1.6.1 six-1.10.0

$ pip install django-suit
Collecting django-suit
  Downloading django-suit-0.2.18.tar.gz (587kB)
Building wheels for collected packages: django-suit
  Running setup.py bdist_wheel for django-suit ... done
  Stored in directory:
/Users/epallier/Library/Caches/pip/wheels/12/8b/9a/e02ab0ad9229881638aa040d47d77c8f562999533811927d41
Successfully built django-suit
Installing collected packages: django-suit
Successfully installed django-suit-0.2.18

- **Install the web application server gunicorn (will be used in production instead of the dev django web server)** :

$ pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-19.4.5-py2.py3-none-any.whl (112kB)
Installing collected packages: gunicorn
Successfully installed gunicorn-19.4.5

- **Install the python mysql client**:

$ pip install mysqlclient
...

  - => Issue under Mac OS X:

$ pip install mysqlclient
Collecting mysqlclient
  Downloading mysqlclient-1.3.7.tar.gz (79kB)
Building wheels for collected packages: mysqlclient
  Running setup.py bdist_wheel for mysqlclient ... error
…

  Failed building wheel for mysqlclient
  Running setup.py clean for mysqlclient
Failed to build mysqlclient
Installing collected packages: mysqlclient
  Running setup.py install for mysqlclient ... done
Successfully installed mysqlclient-1.3.7

BOUH !!!

$ pip install --upgrade wheel
Collecting wheel
  Downloading wheel-0.29.0-py2.py3-none-any.whl (66kB)
Installing collected packages: wheel
  Found existing installation: wheel 0.24.0
    Uninstalling wheel-0.24.0:
      Successfully uninstalled wheel-0.24.0
Successfully installed wheel-0.29.0

```
$ pip uninstall mysqlclient

$ pip install mysqlclient
Collecting mysqlclient
  Using cached mysqlclient-1.3.7.tar.gz
Building wheels for collected packages: mysqlclient
  Running setup.py bdist_wheel for mysqlclient ... done
  Stored in directory:
/Users/epallier/Library/Caches/pip/wheels/9b/06/50/d11418c26cf8f2156b13d4363b5afde8e7e75ebb8540d0228d
Successfully built mysqlclient
Installing collected packages: mysqlclient
Successfully installed mysqlclient-1.3.7

YES !!!
```

- => Issues under Ubuntu:

```
$ pip install mysqlclient
Collecting mysqlclient
  Downloading mysqlclient-1.3.7.tar.gz (79kB)
    100% |□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□| 81kB 1.5MB/s
    Complete output from command python setup.py egg_info:
    /bin/sh: 1: mysql_config: not found
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
    [...]
    ----------------------------------------
Command "python setup.py egg_info" failed with error code 1 in /tmp/pip-build-q6j4inuz/mysqlclient/

BOUH !!!

$ sudo apt-get install libmysqlclient-dev

$ pip install mysqlclient
Collecting mysqlclient
  Using cached mysqlclient-1.3.7.tar.gz
Building wheels for collected packages: mysqlclient
  Running setup.py bdist_wheel for mysqlclient ... error
…
    _mysql.c:40:20: fatal error: Python.h: No such file or directory
     #include "Python.h"
                      ^
    compilation terminated.
    error: command 'x86_64-linux-gnu-gcc' failed with exit status 1

    ----------------------------------------
Command "/home/carens_p/pyros/venv_py35_pyros/bin/python3.5 -u -c "import setuptools,
tokenize;__file__='/tmp/pip-build-k3klv92j/mysqlclient/setup.py';exec(compile(getattr(tokenize, 'open',
open)(__file__).read().replace('\r\n', '\n'), __file__, 'exec'))" install --record /tmp/pip-gz242xxs-record/install-record.txt
--single-version-externally-managed --compile --install-headers
/home/carens_p/pyros/venv_py35_pyros/include/site/python3.5/mysqlclient" failed with error code 1 in
/tmp/pip-build-k3klv92j/mysqlclient/

BOUH !!!

$ sudo apt-get install python3.5-dev

$ pip install mysqlclient

YES !!!
```

Test the project

```
$ cd src/

$ ./manage.py runserver
(or gunicorn pyros.wsgi)
==> http://localhost:8000
...
...
Ctrl-c
```

IV - CONFIGURATION of the Django Back Office (administration interface)

Back Office setup

- Prerequisites in src/pyros/settings.py :
  - INSTALLED_APPS must (at least) contain :
    - django.contrib.admin
    - django.contrib.auth
    - django.contrib.contenttypes
    - django.contrib.sessions
  - MIDDLEWARES must (at least) contain :
    - django.contrib.sessions.middleware.SessionMiddleware
    - django.middleware.common.CommonMiddleware
    - django.contrib.auth.middleware.AuthenticationMiddleware

- At least one 'python manage.py migrate' must have been executed

- Create a superuser for the administration :

```
$ python manage.py createsuperuser
```

- For each app of the project, fill the admin.py file :

```
from django.contrib import admin
from app.models import Model1, Model2

admin.site.register(Model1)
admin.site.register(Model2)
```

Reminder : each application must be registered in the settings.py INSTALLED_APPS variable.

- For each model in models.py, add a '__str__()' method in order to identify the object on the back office. Example :

```
class UserLevel(models.Model):
    name = models.CharField(max_length=45, blank=True, null=True)
    desc = models.TextField(blank=True, null=True)
    priority = models.IntegerField(blank=True, null=True)
    quota = models.FloatField(blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'userlevel'

    def __str__(self):
        return (str(self.name))
```

Naming convention : Use self.name when possible, the creation time/date otherwise. Example :

```
class SiteWatch(models.Model):
    updated = models.DateTimeField(blank=True, null=True)
    lights = models.CharField(max_length=45, blank=True, null=True)
    dome = models.CharField(max_length=45, blank=True, null=True)
    doors = models.CharField(max_length=45, blank=True, null=True)
    temperature = models.FloatField(blank=True, null=True)

    class Meta:
        managed = True
        db_table = 'sitewatch'

    def __str__(self):
        return (str(self.updated))
```

Adaptation of the one-to-many and many-to-many display

- The one-to-many relationships are the following (One.many format) :
  - Schedule.sequences
  - Request.sequences
  - Sequence.albums
  - Album.plans
  - Plan.images
  - Telescope.detectors
  - Detector.filters
  - NrtAnalysis.images
  - Filter.plans
  - Detector.albums
  - UserLevel.users
  - Country.users
  - ScientificProgram.requests
  - User.requests
  - StrategyObs.alerts
  - SequenceType.sequences

- For each "many", create a new class in admin.py just after the imports, following these examples :

For Schedule.sequences, Request.sequences and Sequentype.sequences, we will need :

```
class SequenceInline(admin.TabularInline):
    model = Sequence
    fields = ("name",)
    show_change_link = True
```

For Sequence.albums and Detector.albums, we will need :

```
class AlbumInline(admin.TabularInline):
    model = Album
    fields = ("name",)
    show_change_link = True
```

For StrategyObs.alerts, we will need :

```
class AlertInline(admin.TabularInline):
    model = Alert
    fields = ("request.name",) # there is no 'name' attribute in the Alert model
    show_change_link = True
```

- For each "One", declare a new class in admin.py, just after the "Inlines" class declaration, as done in the following examples :

For Request.sequences :

```
class RequestAdmin(admin.ModelAdmin):
    inlines = [
    SequenceInline,
    ]
```

For Detector.filters and Detector.albums :

```
class DetectorAdmin(admin.ModelAdmin):
    inlines = [
    FilterInline,
        AlbumInline,
    ]
```

- The many-to-many relationships are the following :
  - ScientificProgram - User
  - ScheduleHistory - Sequence

- For each many-to-many relationship, declare a new "Inline" class in admin.py just after the imports, like this :

For ScientificProgram - User :

```
class UserAndSPInline(admin.TabularInline):
    model = ScientificProgram.users.through
```

For ScheduleHistory - Sequence

```
class SequenceAndSHInline(admin.TabularInline):
    model = ScheduleHistory.sequences.through
```

Note : The order in the line "model = ScientificProgram.users.through" is very important : the first model (ScientificProgram) is the one in which is declared the ManyToManyField relationship.

- For each many-to-many relationship, declare two new classes in admin.py, just after the inlines, like in the following examples :

For the ScheduleHistory - Sequence relationship :

```
class ScheduleHistoryAdmin(admin.ModelAdmin):
    inlines = [
        SequenceAndSHInline,
    ]
    exclude = ('sequences',) # ScheduleHistory declares the ManyToManyField, and we want to replace its display in the back office, so we won't display the default field

class SequenceAdmin(admin.ModelAdmin):
    inlines = [
        AlbumInline,        # This is the Inline for the one-to-many relationship Sequence.albums
        SequenceAndSHInline,
    ]
```

For the ScientificProgram - User relationship :

```
class ScientificProgramAdmin(admin.ModelAdmin):
    inlines = [
        RequestInline,
    UserAndSPInline,
    ]
    exclude = ('users',) # Same as ScheduleHistory

class UserAdmin(admin.ModelAdmin):
    inlines = [
    RequestInline,   # This is the Inline for the one-to-many relationship User.requests
        UserAndSPInline,
    ]
```

- For each ModelAdmin class in the admin.py, change the registering line

admin.site.register(Album)

to

admin.site.register(Album, AlbumAdmin)

V - INSTALLATION FROM THE BEGINNING (for dev only, history of the initial project creation)

[[pyros_install_from_start|Pyros installation from the beginning]]

Fichiers

pyros_create.sql	25,717 ko	18/03/2016	Etienne Pallier
PYROS_PDM_v021.png	306,365 ko	18/03/2016	Etienne Pallier