i5k_doc Documentation

Release 1.0

Fish Lin

Jul 23, 2018

Table of Contents

This is my introduction to this project

Requirements

My project depend on xxx

Contents:

CHAPTER 1

Pre-requisites

1.1 Python modules

Described in requirements.txt

- Django==1.8.12
- Markdown==2.6.6
- celery==3.1.23
- cssmin==0.2.0
- django-pipeline==1.6.8
- django-simple-captcha==0.4.5
- djangorestframework==2.3.4
- django-rest-swagger==0.3.5
- docutils==0.12
- jsmin==2.0.11
- pillow==2.2.2
- psycopg2==2.6
- pycrypto==2.6.1
- python-memcached==1.57
- python-social-auth==0.2.16
- requests-oauthlib==0.6.1
- wsgiref
- django-suit==0.2.18

1.2 Service-side pre-requisites

- RabbitMQ
- mod_wsgi
- PostgreSQL

CHAPTER 2

Setup guide

This setup guide is tested in Centos 6.7 and django 1.8.12

Note: The following variables may be used in path names; substitute as appropriate:

```
<user>      :  the name of the user doing a set up.
<user-home> :  the user's home directory, e.g., /home/<user>
<app-home>  :  the root directory of the i5K application, e.g., /app/local/i5k
<virt-env>  :  the root directory of the virtualenv this set up creates.
<git-home>  :  the directory containing the django-blast git repository, e.g. <user-
↪home>/git
```

2.1 Project Applications

Clone or refresh the django-blast project:

```
git clone https://github.com/NAL-i5K/django-blast

# Or if the django-blast repository exists:
cd <git-home>
git fetch
```

2.2 Yum

Generate metadata cache:

```
yum makecache
```

2.3 Python 2.7.8

Install necessary packages:

```
sudo yum -y groupinstall "Development tools"
sudo yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel
sudo yum -y install readline-devel tk-devel gdbm-devel db4-devel libpcap-devel xz-
↪devel python-devel
```

Install python 2.7.8 from source:

```
cd <user-home>
wget http://www.python.org/ftp/python/2.7.8/Python-2.7.8.tar.xz
tar -xf Python-2.7.8.tar.xz

# Configure as a shared library:
cd Python-2.7.8
./configure --prefix=/usr/local --enable-unicode=ucs4 --enable-shared LDFLAGS="-Wl,-
↪rpath /usr/local/lib"

# Compile and install:
make
sudo make altinstall

# Update PATH:
export PATH="/usr/local/bin:$PATH"

# Checking Python version (output should be: Python 2.7.8):
python2.7 -V

# Cleanup if desired:
cd ..
rm -rf Python-2.7.8.tar.xz Python-2.7.8
```

Install pip and virtualenv:

```
wget https://bootstrap.pypa.io/ez_setup.py
sudo /usr/local/bin/python2.7 ez_setup.py

wget https://bootstrap.pypa.io/get-pip.py
sudo /usr/local/bin/python2.7 get-pip.py

sudo /usr/local/bin/pip2.7 install virtualenv
```

Build a separate virtualenv:

```
# Make root dir for virtualenv and cd into it:
cd django-blast

# Create a virtual environment called py2.7 and activate:
virtualenv py2.7
source py2.7/bin/activate
```

2.4 Python Modules and Packages

Install additional Python packages:

```
cd <virt-env>

# Cut, paste and run the following bash script.
# If any installation fails script halts:
for package in                          \
    "django==1.8.12"                    \
    "markdown==2.6.6"                   \
    "cssmin==0.2.0"                     \
    "django-pipeline==1.6.8"            \
    "djangorestframework==2.3.4"        \
    "django-rest-swagger==0.3.5"        \
    "django-suit==0.2.18"               \
    "django-axes"                       \
    "docutils==0.12"                    \
    "jsmin==2.0.11"                     \
    "pycrypto==2.6.1"                   \
    "python-memcached==1.57"            \
    "python-social-auth==0.2.16"        \
    "requests-oauthlib==0.6.1"          \
    "wsgiref==0.1.2"                    \
    "pillow==2.2.2"                     \
    "django-simple-captcha==0.4.5"
do
    echo -e "\nInstalling $package..."
    if ! yes | pip install $package ; then
        echo -e "\nInstallation of package $package FAILED"
        break
    fi
done
```

2.5 RabbitMQ

Install RabbitMQ Server:

```
cd <user-home>

# Install RHEL/CentOS 6.8 64-Bit Extra Packages for Enterprise Linux (Epel).
# The 6.8 Epel caters for CentOS 6.*:
wget https://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
sudo rpm -ivh epel-release-6-8.noarch.rpm

# Install Erlang:
sudo yum -y install erlang

# Install RabbitMQ server:
sudo yum -y install rabbitmq-server

# To start the daemon by default when system boots run:
sudo chkconfig rabbitmq-server on

# Start the server:
```

(continues on next page)

(continued from previous page)

```
sudo /sbin/service rabbitmq-server start

# Clean up:
rm epel-release-6-8.noarch.rpm
```

2.6 Celery

Install celery in the virtualenv and configure:

```
# At this point <virt-env> has all project files
# including celery config files.
cd <virt-env>
pip install celery==3.1.23

# Copy files:
sudo cp celeryd /etc/init.d
sudo cp celerybeat /etc/init.d
sudo cp celeryd.sysconfig /etc/default/celeryd
sudo cp celerybeat.sysconfig /etc/default/celerybeat

# Sudo edit '/etc/default/celeryd' as follows:
CELERYD_CHDIR="<virt-env>"
CELERYD_MULTI="<virt-env>/py2.7/bin/celery multi"

# Sudo edit '/etc/default/celerybeat' as follows:
CELERYBEAT_CHDIR="<virt-env>"
CELERY_BIN="<virt-env>/py2.7/bin/celery"

# Set as daemon:
sudo chkconfig celeryd on
sudo chkconfig celerybeat on
```

2.7 Memcached

Install and activate memcached:

```
sudo yum -y install memcached

# Set to start at boot time:
sudo chkconfig memcached on
```

2.8 Database

Install PostgreSQL:

```
# Add line to yum repository:
echo 'exclude=postgresql*' | sudo tee -a /etc/yum.repos.d/CentOS-Base.repo

# Install the PostgreSQL Global Development Group (PGDG) RPM file:
```

(continues on next page)

(continued from previous page)

```
sudo yum -y install http://yum.postgresql.org/9.5/redhat/rhel-6-x86_64/pgdg-centos95-
↪9.5-2.noarch.rpm

# Install PostgreSQL 9.5:
sudo yum -y install postgresql95-server postgresql95-contrib postgresql95-devel

# Initialize (uses default data directory: /var/lib/pgsql):
sudo service postgresql-9.5 initdb

# Startup at boot:
sudo chkconfig postgresql-9.5 on

# Control:
# sudo service postgresql-9.5 <command>
#
# where <command> can be:
#
#     start   : start the database.
#     stop    : stop the database.
#     restart : stop/start the database; used to read changes to core configuration␣
↪files.
#     reload  : reload pg_hba.conf file while keeping database running.

# Start:
sudo service postgresql-9.5 start

#
#  (To remove everything: sudo yum erase postgresql95*)
#

# Create django database and user:
sudo su - postgres
psql

# At the prompt 'postgres=#' enter:
create database django;
create user django;
grant all on database django to django;

# Connect to django database:
\c django

# Create extension hstore:
create extension hstore;

# Exit psql and postgres user:
\q
exit

# Config in pg_hba.conf:
cd <virt-env>
export PATH=/usr/pgsql-9.5/bin:$PATH

# Restart:
sudo service postgresql-9.5 restart

# Install pycopg2:
```

(continues on next page)

(continued from previous page)

```
pip install psycopg2==2.6
```

2.9 Migrate Schema to to PostgreSQL

Run migrate:

```
cd <virt-env>
python manage.py migrate
```

2.10 Apache

Please note: It is essential that tcp port 80 be open in your system. Sometimes the firewall may deny access to it. Check if iptables will drop input packets in the output of this command:

```
sudo iptables -L
```

If you see "INPUT" and "DROP" on the same line and no specific ACCEPT rule for tcp port 80 chances are web traffic will be blocked. Ask your sysadmin to open tcp ports 80 and 443 for http and https. Alternatively, check this iptables guide.

Install Apache and related modules:

```
sudo yum -y install httpd httpd-devel mod_ssl
```

Give the system a fully qualified domain name (FQDN) if needed:

```
# Find out the system IP addres with 'ifconfig'.
# Assuming it is a VM created by Vagrant, this could be 10.0.2.15.
# Sudo edit '/etc/hosts' and add an address and domain name entry.
# For example:
10.0.2.15  virtualCentOS.local virtualCentOS

# Sudo edit the file /etc/httpd/conf/httpd.conf,
# and set the ServerName, for example:
ServerName virtualCentOS.local:80

# Set to start httpd at boot:
sudo chkconfig httpd on

# Check this setting if you wish, with:
sudo chkconfig --list httpd

# Control:
#    sudo apachectl <command>
# Where <command> can be:
#     start         : Start httpd daemon.
#     stop          : Stop httpd daemon.
#     restart       : Restart httpd daemon, start it if not running.
#     status        : Brief status report.
#     graceful      : Restart without aborting open connections.
```

(continues on next page)

(continued from previous page)

```
#     graceful-stop : stop without aborting open connections.
#
# Start httpd daemon:
sudo apachectl start

# Test Apache:
# If all is well. This command should produce copious
# HTML output and in the first few lines you should see:
#   '<title>Apache HTTP Server Test Page powered by CentOS</title>'
curl localhost

# You can also view the formatted Apache test page in your
# browser, e.g., firefox http://<setup-machine-ip-address>
```

CHAPTER 3

Workplace Apps

3.1 Blast

Introduction

I5K BLAST Tutorial is on https://i5k.nal.usda.gov/content/blast-tutorial

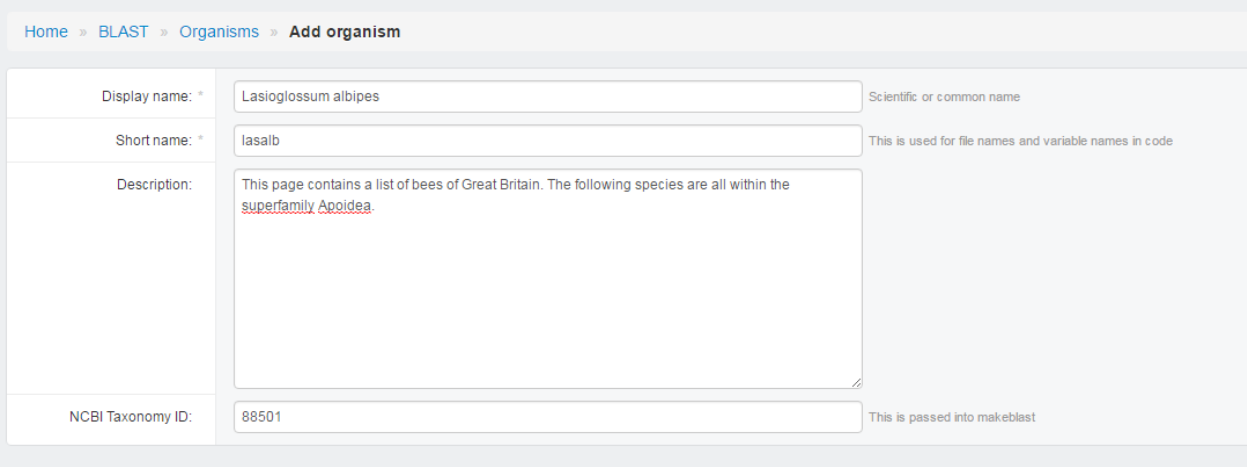
3.1.1 Install & Configuration

Install BLAST and append Blast_bin directory in environment variable `PATH`.

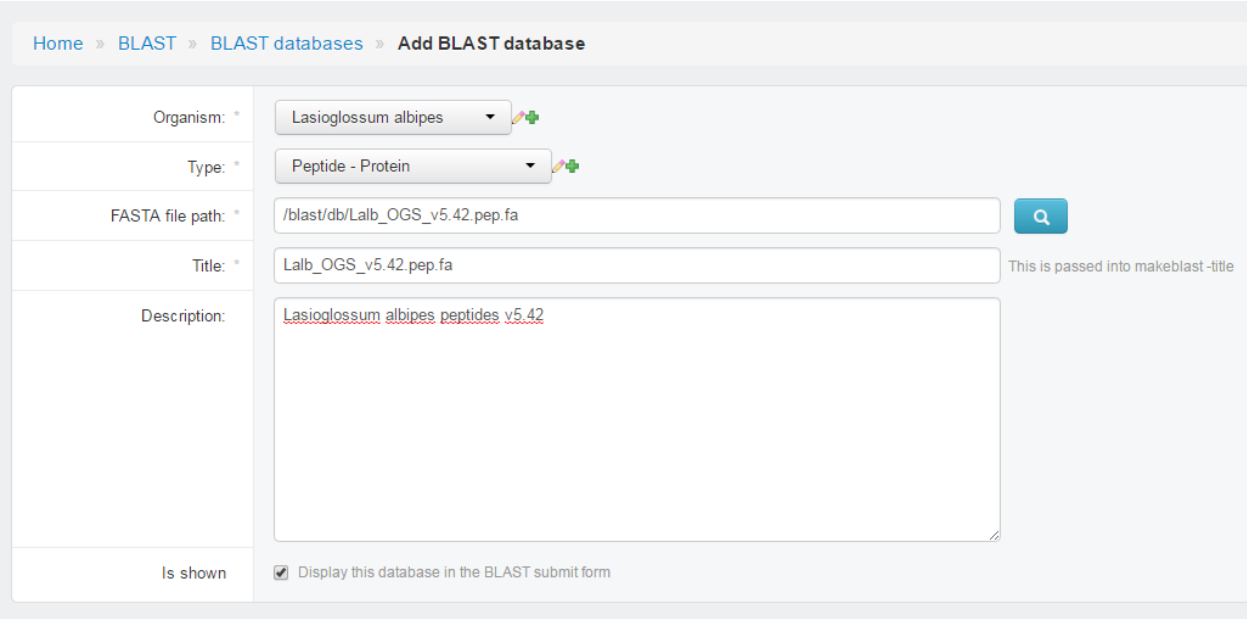
3.1.2 BLAST DB Configuration

There are five tables for creating BLAST DB and browsing in I5K-blast.

- Add Organism:
  - Display name should be scientific name.
  - Short name are used by system as a abbreviation.
  - Descriptions and NCBI taxa ID are automatically filled.



- Add Sequence types:
  - Used to classify BLAST DBs in distinct catagories.
  - Provide two kinds of molecule type for choosing, Nucleotide/Peptide.
- Add Sequence
- Add BLAST DB
  - Choose `Organsim`
  - Choose `Type` (Sequence type)
  - Type location of fasta file in `FASTA file path`
  - Type `Title` name. (showed in HMMER page)
  - Type `Descriptions`.
  - Check `is shown`, if not check, this database would show in HMMER page.
  - Save



- Add JBrowse settings

3.2 Hmmer

HMMER is used for searching sequence databases for homologs of protein sequences, and for making protein sequence alignments. It implements methods using probabilistic models called profile hidden Markov models (profile HMMs).

I5K HMMER Tutorial is on https://i5k.nal.usda.gov/webapp/hmmer/manual.

3.2.1 Install & Configuration

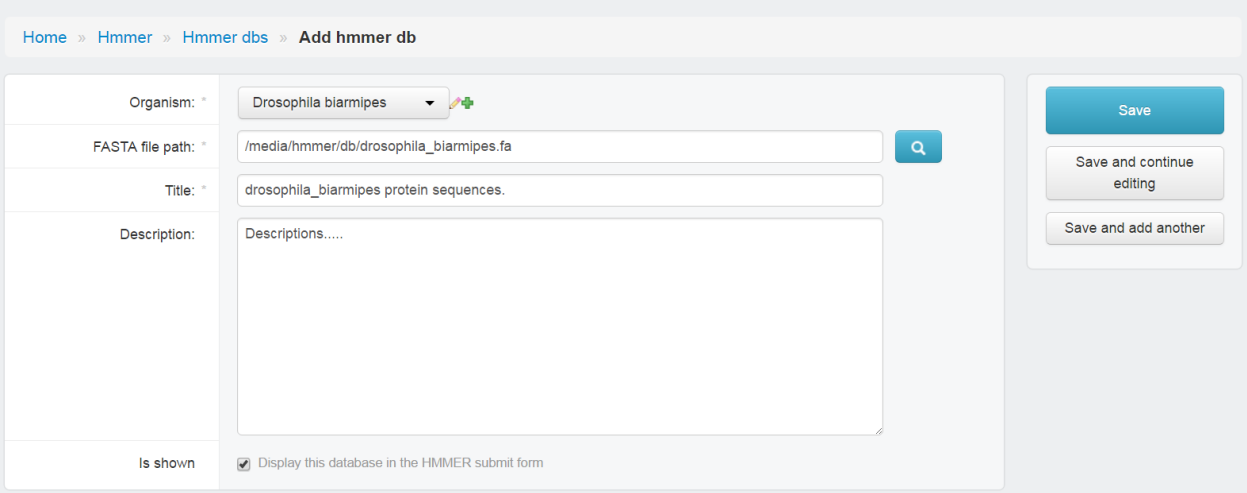
Install HMMER and append HMMER_bin directory in evironment varialbe `PATH`.

3.2.2 HMMER DB Configuration

Like Blast, HMMER databases must be configured then they could be searched.

Go django admin page and click Hmmer on left-menubar. You need to create HMMER db instance (Hmmer dbs) for each fasta file.

- Choose `Organsim`
- Type location of peptide fasta file in `FASTA file path`
- Type `Title` name. (showed in HMMER page)
- Type `Descriptions`.
- Check `is shown`, if not check, this database would show in HMMER page.
- Save



3.2.3 HMMER Query Histroy

HMMER query histories are stored in table `HMMER results`. Users could review them on dashboard. All query results (files on disk) will be removed if it's expired. (default: after seven days)

Query results locate in directory `$MEDIA_ROOT/hmmer/task/`.

3.3 Clustal

ClustalW is the oldest of the currently most widely used programs for multiple sequence alignment. Clustal Omega is the latest version of CLUSTAL series. ClustalO is faster and more accurate because of new HMM alignment engine.

I5K CLUSTAL Tutorial is on https://i5k.nal.usda.gov/webapp/clustal/manual.

3.3.1 Install & Configuration

Install Clustalw and Clustal Omega. Then append both bin directory in evironment varialbe `PATH`.

3.3.2 Clustal Query Histroy

Clustal query histories are stored in table `Clustal results`. Users could review them on dashboard. All query results (files on disk) will be removed if it's expired. (default: after seven days)

Query results locate in directory `$MEDIA_ROOT/clustal/task/`.

3.4 Dashboard

Personal query history.

3.5 Data

Rest framework. Not finished

3.6 Proxy

For providing indirect access to some resources without https. Currently it is used by Web Apollo instances for looking up GO Terms.

3.7 Drupal_SSO

Coonection to Drupal summit data function.

```
DRUPAL_URL = 'https://gmod-dev.nal.usda.gov'

# cookie can be seen in same domain
DRUPAL_COOKIE_DOMAIN=".nal.usda.gov"
```

3.8 WebApollo SSO

Complete introduction locate in Section 4.

CHAPTER 4

WebApollo Single Sign On

4.1 What is WebApollo SSO?

The basic idea in SSO is to provide handy user interface and make WebApollo user more like a community. In order to accomplish those ideas, we try to transfer management jobs from WebApollo to SSO. SSO gives the coordinators more authority to manage their members who can annotating and grant the priviledges on their own.

In SSO, we seperate users into three different roles.

- First, the `ADMIN` who actually owns 'admin priviledge' in WebApollo, can manage users/groups/eroll event.
- Second, the `COORDINATOR` who belong to group `GROUP_(Organism_short_name(OSN))_ADMIN`, can manage membership in specific (Organism).
- Last, the remaining users are in `USER`. They can make request to join (or leave) different organism team. Once be recuited in, user will pertain to group `GROUP_(OSN)_USER`.

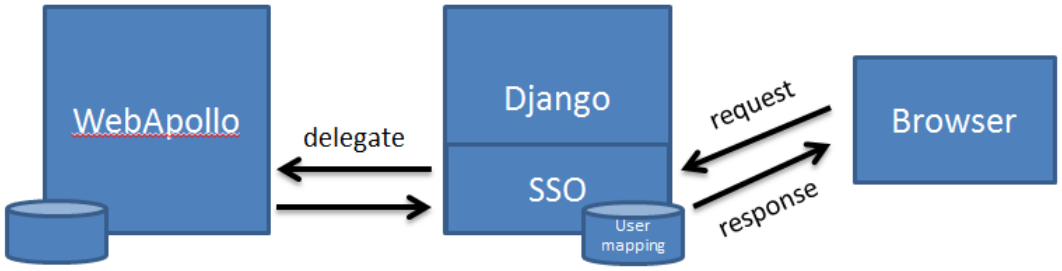
SSO make a virtual role `COORDINATOR` by exploiting a conventional group name `GROUP_(OSN)_ADMIN` and the user in the team would be in group `GROUP_(OSN)_USER`.

Role\	WebApollo	Single Sign On (SSO)
ADMIN	Global Admin	Global Admin
COORDINATOR	Admin permission	in GROUP_()_ADMIN
USER	RWE permission	in GROUP_()_USER with RWE permission

Note: Mapping between full organism name and short organism name are stored in django-blast app. Full organism name is the real name in WebApollo and short name is a abbreviation used in django-blast app.

4.2 Framework Overview

SSO was implemented in Django and JQuery. Conceptually, SSO is a proxy service for delegating user request to appropriate WebApollo service. The main advantage here is that SSO could provides more social utilities for the I5K community.



Database Schema (UserMapping)

Apollo_user_id	Apollo_user_name	Apollo_user_pwd	django_user	last_date
1	Chris	(AES encrpted pwd)	Christopher	
2	Monica	(AES encrpted pwd)	Monica	
3	Mei	(AES encrpted pwd)	NULL	

SSO records the mapping between Apollo_user and django_user in table `UserMapping`. `Apollo_user_id` and `django_user` are unique attribute and this makes mapping a one to one relationship. (apollo_user_name could be changed and is not unique)

In above table, record 1 and 2 tell a formal relationship but record 3 describes an Apollo user doesn't belong to any django user. User can claim it by re-register process. (mentioned below)

4.3 Configuration

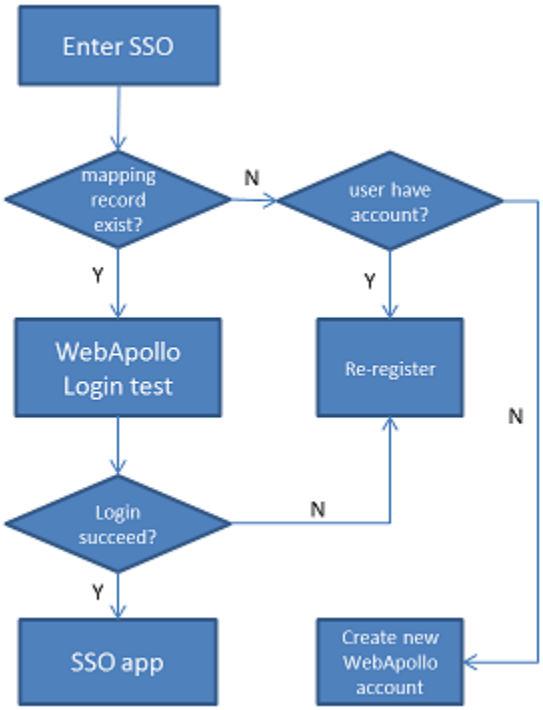
SSO uses a pre-assigned admin Apollo account to communiate with Apollo server. The account must be create on apollo server first. Two URLs address of i5k server and apollo server are used to identify each others' locations. In order to secure user password, SSO encrpt it before saving password into database.

WebApollo SSO configuration in django setting.py:

```
# WebApollo SSO robot account
ROBOT_ID='R2D2'
ROBOT_PWD='demo'

#URL of i5k workspace and webapollo
I5K_URL='http://i5k.nal.gov'
APOLLO_URL='http://i5k.apollo.nal.gov/apollo'

# cookie can be seen in Apollo-prod and Gmod-prod
APOLLO_COOKIE_DOMAIN=".nal.usda.gov"

#Encypt webapollo user password in SSO database.
#AES key must be either 16, 24, or 32 bytes long.
SSO_CIPHER='1234567890123456'
```

4.4 Register WebApollo

There are three ways to make connection between i5k account to apollo account.



- When registering an new i5k account, SSO also create an apollo account(same ID).
- When entering SSO, if SSO doesn't have mapping record of user,
  - it asks user to create a new apollo account
  - or register his account info into SSO.
- When entering SSO, if SSO has mapping record of user but login failed, it asks user to re-enter his password into SSO.

4.5 Utilities

There are six individual tab pages, three of them are general and others are specific for Admin user.

4.5.1 Utilities only for Admin

Tab\	Function Descriptions
User(Admin)	View/Create/Delete/Update/Disconnect Apollo User
Group(Admin)	View/Create/Delete Apollo Group
PReq(Admin)	View Pending request

4.5.2 General Utilities

Tab\	Function Descriptions
My Organism	Manage organism which you joined in / Go WebApollo
My Request	Make request to join/leave a organism community
My Info	User basic information

CHAPTER 5

About i5k Workplace

5.1 About i5k

we are i5k group

5.2 Contact

xxxx

CHAPTER 6

Indices and tables

- genindex
- modindex
- search