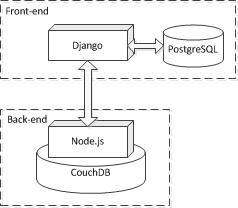
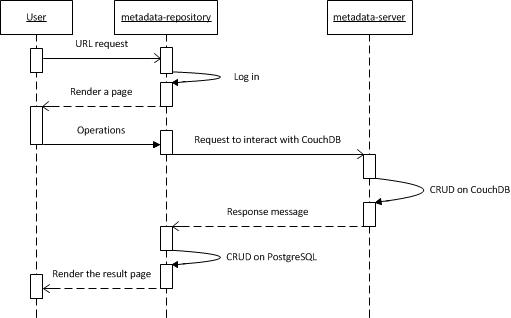
Architecture

* Two tiers of the repository system:



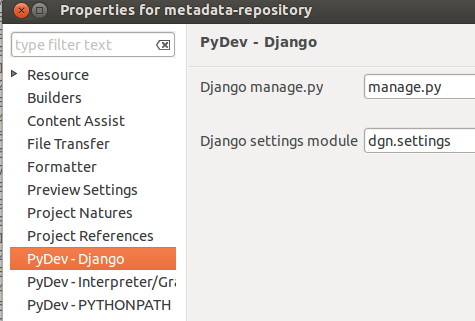
* Front-end: **metadata-repository**
* used to manage authentication information for the repository system and render UI
* designed in Django framework
* Back-end: **metadata-server**
* used to interact with CouchDB database
* designed on Node.js platform
* Workflow diagram:



Metadata-repository

Location at Github: <https://github.com/usgin/metadata-repository>

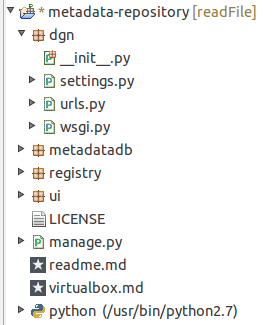
* Prepare the development:
* Prerequisites:
* Python
* Django
* PostgreSQL
* Required external Python modules and their dependencies:
* BeautifulSoup: easy\_install beautifulsoup4
* pyjade: easy\_install pyjade
* lxml: easy\_install lxml
* poster: easy\_install poster
* django-recaptcha: easy\_install django-recaptcha
* Configure Eclipse IDE:
* In ‘Project’ menu, configure ‘Properties’ for this project, e.g.



* In ‘Main’ tab of ‘Debug Configurations’, set ‘Main Module’ as the path to ‘manage.py’, e.g.

${workspace\_loc:metadata-repository/manage.py}

* In ‘Arguments’ tab of ‘Debug Configurations’, set ‘Program arguments’ as ‘runserver --noreload 0.0.0.0:8888’ (Optional, it makes your host machine able to access the Django page in your VM)
* Working on the local machine
* Set up the ‘metadata-repository’ project:
* Create a Django project in your Eclipse IDE
* Clone [https://github.com/usgin/metadata-repository](https://github.com/usgin/metadata-repository%20) and merge the repository with the project. The default ‘urls.py’ has to be replaced with the ‘url.py’ in ‘metadata-repository/dgn’. The project directory will look like:



* Branches:
* ‘readFile’ branch: currently is in use for respository.stategeothermaldata.org
* ‘master’ branch: without file upload function
* ‘unAuthenFunc’ branch: with login-not-required urls based on ‘master’ branch (https://github.com/usgin/metadata-repository/blob/unAuthenFunc/metadatadb/readme.md)
* The module ‘metadatadb’:

It includes urls interacting with ‘metadata-server’ and PostgreSQL. The urls are defined in ‘metadatadb/urls.py’. All the functions are saved in the folder ‘metadatadb/proxy’.

For example, in order to implement url ‘/metadata/upload/’:

* Insert ‘url(r'^upload/$', 'uploadRecord')’ to ‘metadatadb/urls.py’
* Insert ‘from harvestRecord import uploadRecord’ to the end of ‘metadatadb/proxy/\_\_init\_\_.py’
* Insert a new function ‘uploadRecord’ to ‘metadatadb/proxy/harvestRecord.py’
* The module ‘ui’:

It includes urls used to render JavaScript pages. All JavaScript files are written using CoffeeScript, and saved in folder ‘ui/coffee’.

Run ‘cake build’ in directory ‘ui’ to compile CoffeeScript into JavaScript.

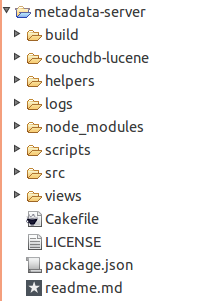
* Update the production server ‘repository.stategeothermaldata.org’:
* The path is ‘~/web/metadata-repository’
* If the static files are modified, change directory to ‘web/usgin\_django’. Then run python manage.py collectstatic
* Restart Django server: sudo restart usgin\_django

Metadata-server

Location at Github: <https://github.com/usgin/metadata-server>

Check the ‘[readme.md](https://github.com/usgin/metadata-server/blob/master/readme.md)’ document for instruction

Run ‘cake build’ in the root directory to compile CoffeeScript as JavaScript into the ‘build’ folder



* Helpers:

The folder ‘build/helpers’ contains the Node.js files which can be used to interact with CouchDB directly. Write CoffeeScript in ‘helpers’ folder first. Then execute ‘cake build’ to translate CoffeeScript into JavaScript in ‘build/helpers’ folder.

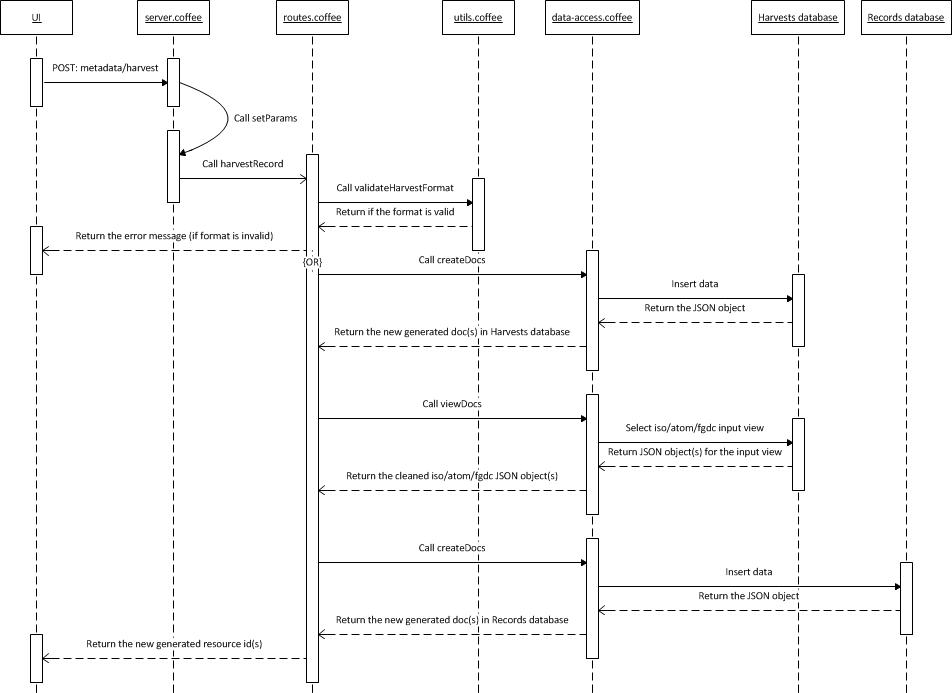
For example, run ‘deleteCollection’ function:

node build/helpers/deleteCollection.js {arg1} {arg2}...

The document ‘[helpers/readme.md](https://github.com/usgin/metadata-server/blob/master/helpers/readme.md)’ demonstrates how to use those functions.

* Workflow - metadata harvest example:

The urls are defined in ‘src/server.coffee’. See the following diagram for a basic request to harvest metadata:



Q&A

* How to modify the Django admin page?

This is a useful document: <https://docs.djangoproject.com/en/dev/ref/contrib/admin/>

For example, if you are going to add a new admin action ‘Mark selected resources as published’, you need to add following codes into ‘registry/admin.py’:

def make\_published(modeladmin, request, queryset):

queryset.update(published=True)

make\_published.short\_description = "Mark selected resources as published"

Then add ‘actions = [make\_published]’ in function ‘ResourceAdmin(admin.ModelAdmin)’

* How to publish all the records in a given collection?

You have to publish the records in Django and CouchDB separately.

For example, if you are going to publish all the records in collection ‘AASG Geothermal Data Project’:

1. Find the collection id ‘9e15e1a59b768b330d029e86dc007dbc’ for ‘AASG Geothermal Data Project’.
2. Select all the records in this collection in the Django admin page. Then publish them.
3. Log into the remote server ‘repository.stategeothermal.org’ and change directory to ‘~/web/metadata-repository’.
4. Execute the node.js script ‘publishCollection’ to publish all records in the collection ‘AASG Geothermal Data Project’

node build/helpers/publishCollection.js “9e15e1a59b768b330d029e86dc007dbc”