UNIVERSITÉ DE STRASBOURG

Information Technology Department

CS3 Workshop on Cloud Services for File Synchronisation and Sharing

Scalable management with Seafile SPORE specification

Wednesday, the 1st February 2017

Vincent Lucas - lucas@unistra.fr

Sébastien Finkbeiner - s.finkbeiner@unistra.fr

1. Introduction
Background, problem and goal
Example

2. Idea and contribution
Automate
Delegate

3. Conclusion

Strasbourg University provides a file synchronization and sharing service for:

- 11000 researchers, teachers and employees,
- 900 structures, such as schools, laboratories and departments.

Problem:

- Start: 15 GB standard account for everyone.
- End: customisation requests for teams, projects, schools, etc.

At this scale, a centralized management is not an option.

Goal:

- *automate* and *delegate* management for:
  - accounts, guest accounts,
  - groups,
  - shared repositories
  - and space quotas.

Request:

> *"Laboratory α needs λ TB for a new WizWoz project."*

Required actions:

- Create a dedicated *WizWoz* account.
- Allocate λ TB to the *WizWoz* account.
- Create a dedicated group:
  - Populate it with the project user list.
  - Create guest accounts for external contributors.
- Create a dedicated project repository:
  - Host it on the *WizWoz* account.
  - Share it to the group.

Solution 1:

- Administrator web interface
- Konami code: Clic, clic, type, clic ... type, clic and clic.

Pros:

- It works out of the box.
- Perfect for tiny configuration correction.

Cons:

- Takes a lot of time.
- Increases error ratio.
- Can not be reproduced from development to production platform.
- Centralized: needs an administrator account.

Automate: ✗
Delegate: ✗

Solution 2:

- Web API

Pros:

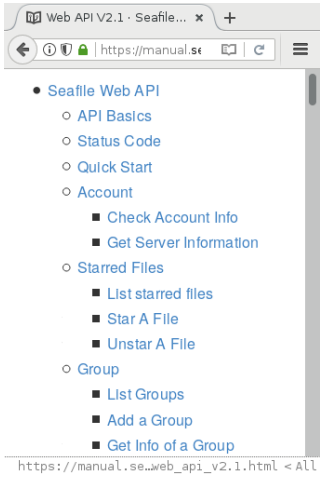
- Well documented:
  `https://manual.seafile.com` ⇒
- Easy to try and validate via:
  - web interface
  - curl

Cons:

- Some work to achieve our goal:
  - automate and delegate

Leads to:

- Solution 2++: SPORE



Solution 2:

- Web API

Pros:

- Well documented:
  `https://manual.seafile.com`
- Easy to try and validate via:
  - web interface ⇒
  - curl

Cons:

- Some work to achieve our goal:
  - automate and delegate

Leads to:

- Solution 2++: SPORE

Solution 2:

- Web API

Pros:

- Well documented:
  `https://manual.seafile.com`
- Easy to try and validate via:
  - web interface
  - curl ⇒

Cons:

- Some work to achieve our goal:
  - automate and delegate

```
#$ curl -D - http://10.0.3.13/api2/ping/
HTTP/1.1 200 OK
Server: gunicorn/19.4.5
Date: Sun, 29 Jan 2017 13:08:18 GMT
Connection: close
Transfer-Encoding: chunked
Vary: Accept, Accept-Language, Cookie
Content-Type: application/json
Content-Language: en
Allow: GET, HEAD, OPTIONS

"pong"
```

Leads to:

- Solution 2++: SPORE

1. Introduction
Background, problem and goal
Example

2. Idea and contribution
Automate
Delegate

3. Conclusion

What is SPORE?

- A web service description.
- Saved in a simple JSON document.

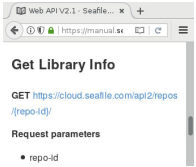
Why SPORE?

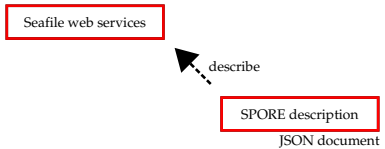
- Dynamically generate high level client objects.
- Implementations available:
  - Clojure, Javascript, Lua, Nodejs, Perl, Python, Ruby, Livescript and Groovy.
- Strasbourg University best practice to use a web service.

Most important:

- Makes your web service reusable

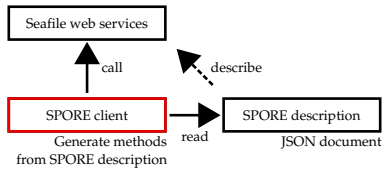
Seafile web services:





SPORE description:

```
"repo": {
        "path": "/api2/repos/:repo_id",
        "required_params": [
                "repo_id"
        ],
        "expected_status": ["200"],
        "method": "GET",
        "authentication": true,
        "description": "Retrieve a library"
},
```



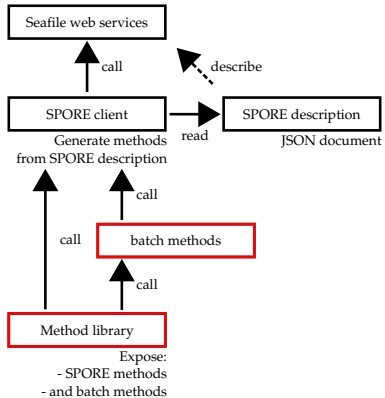

```
#!/usr/bin/python3
import britney # install note: pip3 install britney
from britney.middleware.format import Json

# Generate a SPORE client.
spore_client = britney.new(
    "http://rest-api.u-strasbg.fr/seafile/description.json",
    base_url="https://my_seafile_server_url")
spore_client.enable(Json)

# Authenticate to Seafile instance
conf = {'username': 'admin@example.com', 'password': 'my_pwd'}
result = spore_client.auth_token(payload=conf)
token = result.data["token"]
spore_client.enable('ApiKey', ** {'key_name': 'Authorization','key_value': 'Token ' + token})

# Call the "repo" fonction from description file.
result = spore_client.repo(**{'repo_id': '1234242a-c94c-45d1-9475-1243432'})

print("status: {0}, data: {1}".format(result.status_code, result.data))
```

Batch method example:

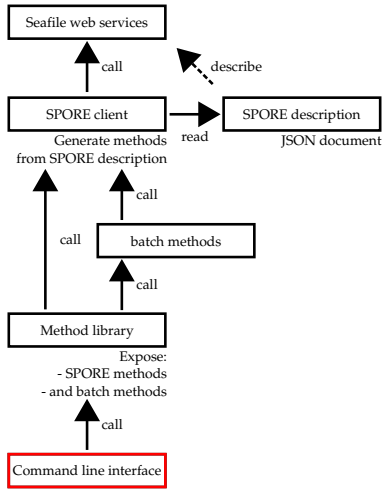
- e.g. create a non-SPORE "create_guest_account" method:
  - "account_info":
    check if the account exists.
  - "account_create":
    create the requesting account.
  - "account_update":
    set the quota to 0.

Method library:

- Exposes SPORE methods:
  "account_create"
- Exposes batch methods:
  "create_guest_account"



```
#$ ./cmd_seafile_exploit.py h
Command list help:
- account_create <email:> [ <is_staff:> <is_active:> ]
        Account creation
- account_delete <email:>
        Account deletion
- account_info <email:>
        Account information
- account_update <email:> [ <password:> <is_staff:> <is_active:>
            <name:> <note:> <storage:> ]
        Account update
- accounts [ <start:> <limit:> <scope:> ]
        List accounts
[...]
- auth_token <username:> <password:>
        Get a authentification token
- b <email:>
        Batch to create users:
            auto generate password and send email.
[...]
- bg <file>
        Batch to create groups:
            auto generate groups and accounts.
        File must follow the syntax:
                group_name,group_owner,rights,user1email,user2email,etc.
[...]
```

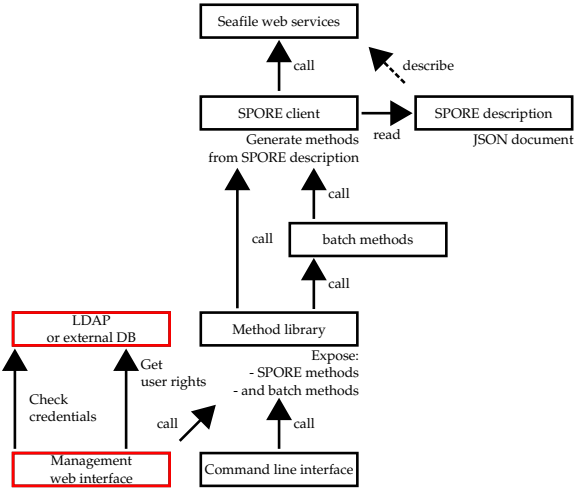


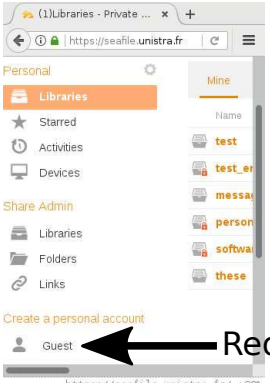
Automate: ✓
Delegate: ✗

Management web interface:

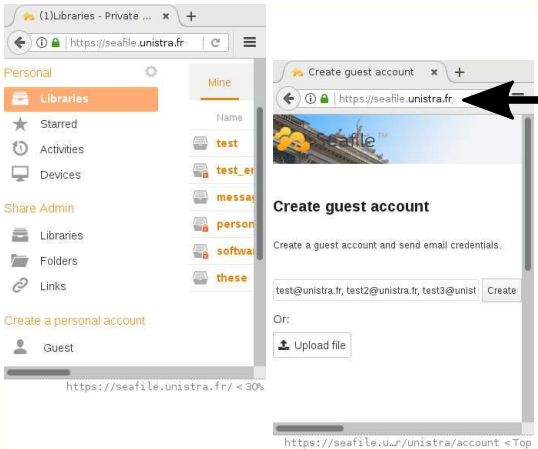
- *UWSGI* web pages
  e.g. *bottle*,
  cf. `http://bottlepy.org`
- Connect to LDAP, database, etc.,
  to get or compute user rights
- Call the method library,
  if the user have the corresponding rights

Without Seafile administrator right,
a user can access (with control)
to privileged Seafile methods.

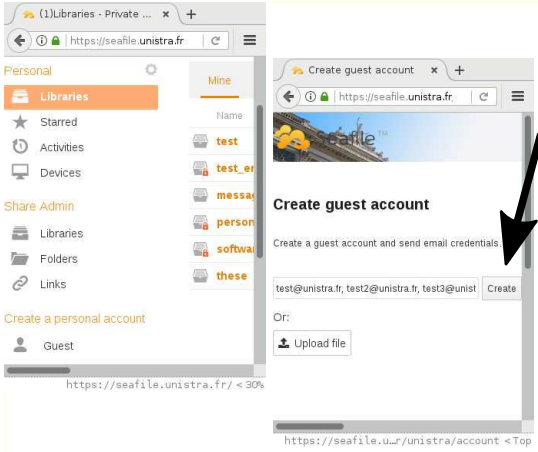




Redirect to UWSGI page



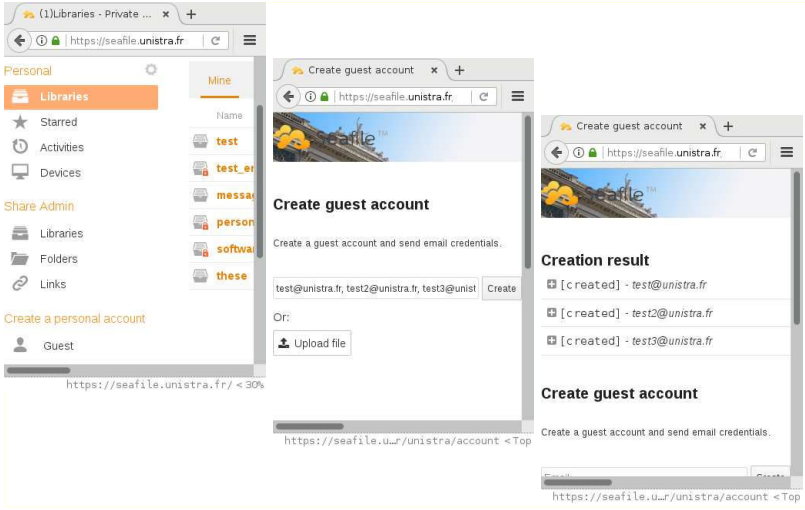
1) Check Django credentials
2) Check LDAP rights
3) Display page for guest accounts creation



1) Call method library:

"create_guest_account":

1.1) check if account exists
1.2) create the account:
1.2.1) generate a password
1.2.2) create an account
1.2.3) send an email with the credentials
1.3) set quota to 0


(1)Libraries - Private ...
https://seafile.unistra.fr
Personal
Libraries
Starred
Activities
Devices
Share Admin
Libraries
Folders
Links
Create a personal account
Guest
Mine
Name
test
https://seafile.unistra.fr/ < 30%
Create guest account
https://seafile.unistra.fr
Create guest account
Create a guest account and send email credentials.
test@unistra.fr, test2@unistra.fr, test3@unist
Create
Or:
Upload file
https://seafile.u...r/unistra/account < Top
Create guest account
https://seafile.unistra.fr
Creation result
[created] - test@unistra.fr
[created] - test2@unistra.fr
[created] - test3@unistra.fr
Create guest account
Create a guest account and send email credentials.
https://seafile.u...r/unistra/account < Top

Automate: ✓
Delegate: ✓

1. Introduction
Background, problem and goal
Example

2. Idea and contribution
Automate
Delegate

3. Conclusion

Finally, this solution achieved to:

- *Automate* management process:
  - *Seafile* web services,
  - SPORE description:
    - publicly available:
      `http://rest-api.u-strasbg.fr/seafile/description.json`
    - ≈ 38% (50/130) methods already described
  - SPORE client:
    - cf. Python example with *britney* in this presentation
      More details available at:
      `https://github.com/agrausem/britney`
    - More clients available at:
      `https://spore.github.io/`
- *Delegate* management process:
  - Bind together SPORE client and external services
  - Computes and delegates user rights.
  - Exposes granted methods via web interface.

Questions?

UNIVERSITÉ DE STRASBOURG

Information Technology Department

CS3 Workshop on Cloud Services for File Synchronisation and Sharing

Scalable management with Seafile SPORE specification

Wednesday, the 1st February 2017

Vincent Lucas - lucas@unistra.fr

Sébastien Finkbeiner - s.finkbeiner@unistra.fr