ASRimbra

A distributed and easily scalable application to exchange message with other user

Project made for a scholar microprojet

Download

```
cd folder/where/you/want/to/download/ASRimbra
git init
git remote add origin git@bitbucket.org:pandawan/asrimbra.git
git pull
```

Configuration

You can configure ASRimbra behavior using the constant defined in the different modules

Installation

You need to have maven and Java JDK 1.8+ installed on your computer first.

Download the dependencies, run tests and install modules with maven : > `mvn install`

It will be faster not to run tests: > `mvn install -DskipTests`

To package DirectoryManager use, *in its subfolder*:

```
mvn package
```

This generated target/ASRimbra-DirectoryManager-{version}-jar-with-dependencies.jar.

To package MailboxManager use, *in its subfolder*: > `mvn package`

This generated target/ASRimbra-MailboxManager-{version}-jar-with-dependencies.jar .

You can launch both managers using, *in their subfolders*: > `mvn exec:java`

Infrastructure

We have a nice infrastructure !

You can find the basic infrastructure at ./ASRimbra_architecture.png

Some choice were made to improve scalability, software evolution and consistency.

Why do we believe than our choices were goods ? Because anyone can do this without having to touch the code: ./ASRimbra_architecture_rev.png

See more at section

Clients

Both ManagerClient and AdministrationClient are web clients. They can be resume to a single static web page having forms used to contact the server-side REST API.

This has several advantages : * It's easy to develop, test and maintain (click on a button to create a request, the browser will soon remember your parameters). * It's easy to send on other devices: * It works in local without dependencies * It works on a server. * It is static ! Nothing is easier to cache. * It works if it is saved from a server (Ctrl + S on the page and TADA ! You have really save the edu.tsp.asr.asrimbra.client, not just a generated file without any logic) * It is easy to create an software from it : it just need to be wrapped in a WebView * It is easy to create a mobile application from it : it just need to be wrapped in a WebView (eg : with Cordova) * It works with a REST API, and a REST API make it easier to interconnect ASRimbra with other services. * Even in command line, you can use it through lynx-like browser. * Browser and WebView in general have no problems to store cookies, and as next section explain, ASRimbra use cookies.

Server complies with browser particular demands on REST API related to Cross Origin Resource Sharing (CORS): Theirs responses will have the request origin, and they provide an Option method where there is a non-GET/POST method available.

Communication and authentication

As said above, ASRimbra use a REST API to communicate with server.

Authentication with token Authentication is managed through a Token.

This token is of the form username/expiration date/signature

Signature is a hash of the username, the expiration date and a key hard coded in all the servers.

This means that once you have your token, any server can checked if you are connected with a simple comparison between the signature of your token and what they calculate from your username, expirationDate and this key.

No sessions are need to perform this check, so the server is fully stateless. **This mean easy scalability.**

No database access are need to perform the check, we have a statelessness at a very low cost.

One inconvenient is that you cannot revoke token, you can only destroy cookie and if someone save the value of it, he can regenerated it and continue to browse even if its account was destroyed until the cookie die. We do not believe it is a problem with a time to live for the cookie of 1 hour.

REST API DirectoryManager default base URI is `http://server:7654`

Following routes are supported by DirectoryManager :

GET "/user/getAll" — List all users

POST "/user/addByCredentials" — Add a user from a mail and a password — Query parameters are "mail", "password"

POST "/user/add" — Add a user from a mail, a passwordHash and a role — Query parameters are "mail", "passwordHash" and "role"

OPTIONS "/user/removeByMail/:mail" — Return the available options for this route

DELETE "/user/removeByMail/:mail" — Delete a user with the mail :mail.

GET "/user/getRight" — get the rights of user which mail is mail. — Query parameter is "mail"

POST "/user/setAdmin" — set Admin the user which mail is mail. — Query parameter is "mail"

POST "/user/setSimpleUser" — set Simple User the user which mail is mail. — Query parameter is "mail"

GET "/user/getByMail" — return data about the user which mail is mail. — Query parameter is "mail"

GET "/user/getRoleByCredentials" — return a Simple User the user which mail is mail. — Query parameter is "mail" "login", "password"

GET "/disconnect" — destroy token in your cookie.

If the route is prefixed by user/, then you have to be authenticated as an Admin to access it.

Following routes are supported by MailboxManager :

MailboxManager default base url is `server:4567`

POST “/connect” — Return a token (after identity check) in a cookie — Query parameters are “mail” and “password”

GET “/disconnect” — destroy token in your cookie.

GET “/mailbox/” — List all the messages associated to you (identity retrieved from token)

GET “/mailbox/:id” — Display message which id is :id if it is associated to you (identity retrieved from token).

DELETE “/mailbox/:id” — Remove message which id is :id if it is associated to you (identity retrieved from token). *

POST “/mailbox/send/” — Send a message from your account to to with title title and content content (identity retrieved from token). — Query parameters are “to” “title” and “content”

If the route is prefixed by user/, then you have to be authenticated to access it.

*: This mean that both the sender and the receiver can delete an information of our servers. It is a choice, both should be able to remove what they have written/receive from our servers.

Servers

Both DirectoryManager and MailboxManager are build on Spark.

It allows us to create complex application with minimal syntax.

Spark server are**multithreaded**and easy to configure.

Entities and persistence

There are 4 entities in ASRimbra: * User * Mail * Role — use to identify admins from simple users (a simple Enum). * MailingList — use for instance for newsletters.

User, Mail and MailingList are managed with repositories.

One of the biggest constraint of the application is that there is no direct access from MailboxManager to User DB and no direct access from MailboxManager to Mail(and MailingList) DB.

- MemoryRepositories does not allow that and does not provide durability, thus they are no longer used but where usefull during development and may be usefull again in certains case (eg tests).
- JPARepositories allow access to a DB with hibernate

- RemoteRepositories allow access to a distant server which has to implement a corresponding REST API. It is used for the communications from MailboxManager to DirectoryManager.

RemoteRepositories are basically a **proxy** for a remote repository. It uses**synchronous call.**Since in all cases the responses is need to go further in the algorithm, asynchronous/buffered messages/inversion of control/etc would not have been really useful.

Distributed ASRimbra

Our goal is to have this: ./ASRimbra_architecture_rev.png

The most important point is that we do not need to change our code : some configurations may not be compatible with this (eg: use of MemoryRepositiories), but the basic one is.

How-to

Add a ningx server Reverse proxy must be accessible from the outside and access to alls internal servers (requires multiple network interface (see `Deploy servers`), may be usefull to create it in a VM).

We use a nginx server to act as a ReverseProxy, LoadBalancer and Firewall.

You can use another server but nginx is very well suited for those tasks.

You need to create 3 entries in site-availables (and enable them): * `dir-man.yourdomain.com` * `mail-man.yourdomain.com` * `dir-api.yourdomain.com`

dir-api.yourdomain.com MUST only be accessible from dir-man.yourdomain.com and mail-man.yourdomain.com servers addresses.

See documentation for having multiple sites and multiple server by sites.

You can choose between several load balancing algorithm but round robin should work fine in most cases.

Note that it can also be used as a content deliver for clients: As said below, clients are static files and nginx is very, very good to deliver static files.

Deploy servers All your servers must be in a private LAN. Change properties in modules so that they use your new urls.

Change DNS configuration You need to create as many entries in your DNS server as there is in : * dir-man.yourdomain.com * mail-man.yourdomain.com * dir-api.yourdomain.com All of them must redirect to your nginx server.

Note that dir-api.yourdomain.com DNS configuration should not be accessible from anywhere else that dir-man.yourdomain.com and mail-man.yourdomain.com. You may want to add it directly to /etc/hosts instead.

And... Voilà ! Nothing else to do, just launch the edu.tsp.asr.asrimbra.client and it works if previous steps were made correctly.

Understanding the code

Java 8 ASRimbra use Java 8's streams, lambdas, optional, etc. You may want to have a look at those concepts before going further. * Streams: http://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/ * Lambda expression: https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html * Optional: http://www.oracle.com/technetwork/articles/java/java8-optional-2175753.html

Spark Framework ASRimbra use Spark Framework for its REST API.

Spark Framework is a java web framework and has nothing to do with Apache Spark which is a computing framework

To be able to understand the code, you may want to read Spark Framework documentation: * http://sparkjava.com/documentation.html

Hibernate ASRimbra use hibernate to communicate with databases (default behaviour)

Information are defined in core/src/main/resources/hibernate*.cfg.xml

- Tutorial for using hibernate and POJO: http://www.tutorialspoint.com/hibernate/index.htm

DavidWebb ASRimbra use DavidWebb, a lightweight Java HTTP-Client for calling JSON REST-Services.

To be able to understand the code, you may want to discover DavidWebb repository: * https://github.com/hgoebl/DavidWebb

PBKDF2 ASRimbra use a PBKDF2 by rtner.de to calculate hash of sensitive information. More on

- More on PBKDF2: https://en.wikipedia.org/wiki/PBKDF2
- More on this implementation: http://www.rtner.de/software/PBKDF2.html