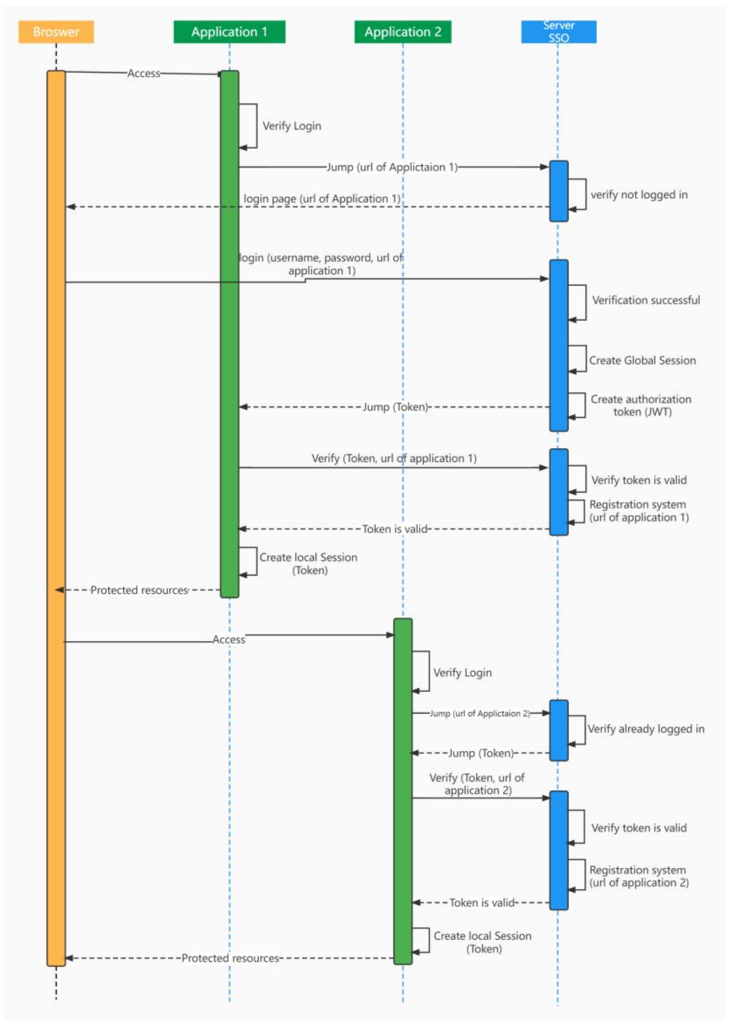
1. Code Repository

Github: https://github.com/EricXiaxl/springboot-sso-jwt.git

2. Tech Stack

- Jdk 8
- Spring/SpringBoot
- Spring Security
- OAuth2
- JWT

3. Sequence Diagram


Broswer
Application 1
Application 2
Server SSO
Access
Verify Login
Jump (url of Applictaion 1)
verify not logged in
login page (url of Application 1)
login (username, password, url of application 1)
Verification successful
Create Global Session
Create authorization token (JWT)
Jump (Token)
Verify (Token, url of application 1)
Verify token is valid
Registration system (url of application 1)
Token is valid
Create local Session (Token)
Protected resources
Access
Verify Login
Jump (url of Applictaion 2)
Verify already logged in
Jump (Token)
Verify (Token, url of application 2)
Verify token is valid
Registration system (url of application 2)
Token is valid
Create local Session (Token)
Protected resources

4. Goals

Goal 1: Design and implement a third-party authorization center service (Server) to complete user login, authentication and authority processing.
Goal 2: Any number of client applications (Client) can be mounted under the authorization authentication center.
Goal 3: When user accesses the security page of the client application, it will be redirected to the authorization center for identity verification. After the authentication is completed, the service of the client application can be accessed, and multiple client applications only need to log in once (Single Sign On).

Driven by this goal, this paper designs three independent services, namely:
a. An authorization authentication service center (**vmware-server**)
b. Client Application 1 (**vmware-client1**)
c. Client Application 2 (**vmware-client2**)

5. SSO - Authorization/Authentication Center (vmware-server)

- AuthorizationServerConfig: (i) define the passports of two client applications (vmware-client1 and vmware-client2); (ii) configure the specific implementation of token as JWT Token.
- SpringSecurityConfig: the configuration of Spring Security to configure exception handlers and set whether the interface requires authentication.
- LoginUserDetailsService: implement the UserDetailsService interface of Spring Security to verify the legitimacy of the current user and query user permissions. (In the current task, temporarily simulate a user, the username is ‘test’, the password is ‘pwd123’, and grant the user 'ROLE_NORMAL' and 'ROLE_MEDIUM' permissions)

6. Client Application (vmware-client1 and vmware-client2)

- ClientWebsecurityConfigurer: Spring Security configuration for client applications. to configure whether interface request requires authentication.
- application.yml: security configuration (oauth2 and JWT) and configuration of the server

address of vmware-server for communicating with the Authorization/Authentication Center.

- TestController: contains three interfaces, which require three permission types (ROLE_NORMAL, ROLE_MEDIUM, ROLE_ADMIN) respectively, to verify interface access control.

7. Details for testing

a. Run Order:

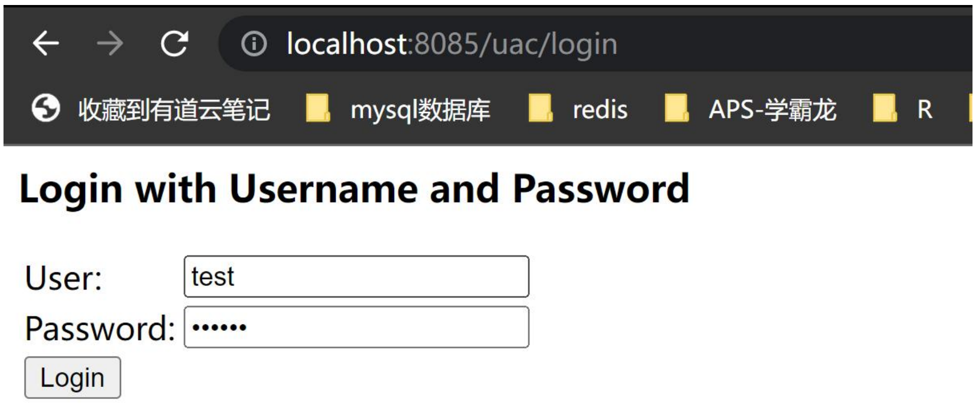
(i) Run the **vmware-server** on the localhost port **8085**

(ii) Run the **vmware-client1** on the localhost port **8086**

(iii) Run the **vmware-client2** on the localhost port **8087**

b. Test Result

At first, using a browser to access the test interface of client application 1 (**vmware-client1**): **localhost:8086/normal**. Since there is no user login authentication at this time, it will automatically jump to the login authentication page of the authorization authentication center: **http:// localhost:8085/uac/login**:



localhost:8085/uac/login

收藏到有道云笔记 mysql数据库 redis APS-学霸龙 R

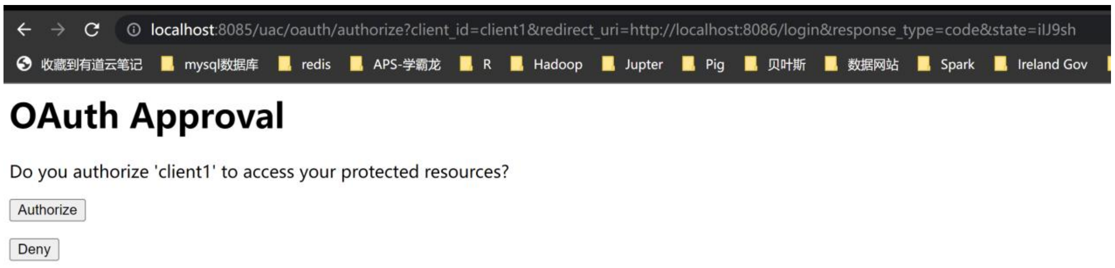
Login with Username and Password

User: test

Password: ••••••

Login

Input the username ‘**test**’ and the password ‘**pwd123**’ to log in for authentication and enter the OAuth authorization page:



localhost:8085/uac/oauth/authorize?client_id=client1&redirect_uri=http://localhost:8086/login&response_type=code&state=ilJ9sh

收藏到有道云笔记 mysql数据库 redis APS-学霸龙 R Hadoop Jupter Pig 贝叶斯 数据网站 Spark Ireland Gov

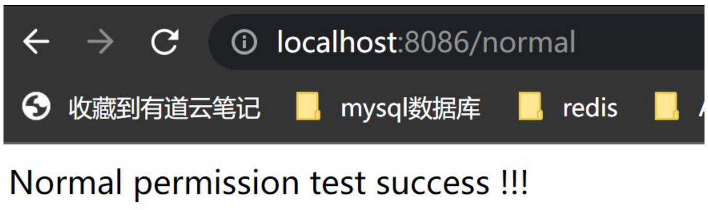
OAuth Approval

Do you authorize 'client1' to access your protected resources?

Authorize

Deny

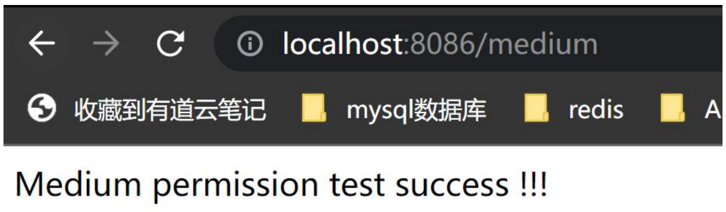
After agreeing to the authorization, it will automatically return to the test interface of the previous client:



localhost:8086/normal

Normal permission test success !!!

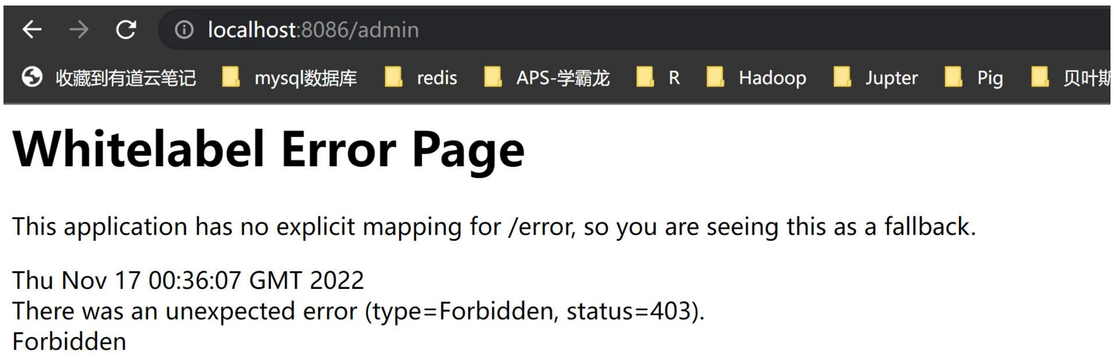
At this point, continuing to request the test interface of client 1 (**vmware-client1**): **localhost:8086/medium**, and find that it can be called directly without authentication:



localhost:8086/medium

Medium permission test success !!!

Since the interface permissions required by **localhost:8086/normal** and **localhost:8086/medium** are available to the user, they can be accessed smoothly. Next, access the interface with higher authority: **localhost:8086/admin.** It can be found that access is forbidden:



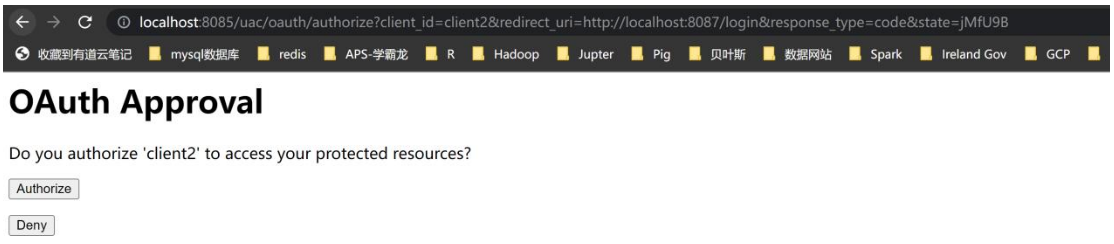
localhost:8086/admin

Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Thu Nov 17 00:36:07 GMT 2022
There was an unexpected error (type=Forbidden, status=403).
Forbidden

After testing access to the interface of client application 1 (**vmware-client1**), then access the test interface of client application 2 (**vmware-client2**): **localhost:8087/normal**, it shows that the access automatically jump to the authorization page without login:

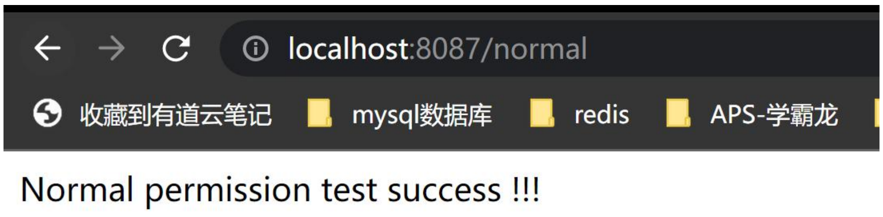


localhost:8085/uac/oauth/authorize?client_id=client2&redirect_uri=http://localhost:8087/login&response_type=code&state=jMfU9B

OAuth Approval

Do you authorize 'client2' to access your protected resources?

After the authorization is completed, it can successfully access the interface of client 2 (**vmware-client2**):



Normal permission test success !!!

The single sign-on (SSO) function has been successfully verified!

8. Future work

Due to time constraints (required less than one hour), this assessment task only practiced the streamlined process: Single Sign-On (SSO) and interface security access control (JWT + Spring Security). There is still a lot of work to be done in the future if I have more time, such as:

(i) Improve the project: cancel the simulated data, write related entity classes (such as User, Role, etc.), and some database operations of the persistence layer, define related interfaces and implementation, and use slf4j to record related logs.

(ii) Using Redis to manage the authentication token, which helps different services to share tokens and obtain user-related information, thereby reducing the frequency of database reads and writes and improving efficiency.

(iii) Better UI for login page of the authorized authentication center.

(iv) Each interface access needs to visit the authentication center, which makes the frequency of network requests between systems too high, resulting in slightly poor efficiency and greater pressure on the authentication center, so it is necessary to build gateway or other micro-service and use RSA encryption algorithm to reduce the pressure on authorization authorization center server.

(v) Customize different exception prompts.