Hướng dẫn sử dụng service Oauth2

I Mở đầu

Tôi xin mở đầu bài viết này bằng một câu chuyện về thực trạng về các ứng dụng hiện nay. Vấn đề đặt ra đó là số lượng ứng dụng đang phát triển chóng mặt, bao gồm các ứng dụng giải trí (nghe nhạc, xem phim, game, ...), ứng dụng đọc tin tức (báo điện tử, blog, ...) và còn nhiều loại ứng dụng khác nữa.

Mỗi ứng dụng như tôi kể phía trên đều có cơ chế quản lý người dùng riêng. Thử tưởng tượng bạn cực kỳ yêu thích khoảng 10-20 ứng dụng, và việc ngồi đăng ký tài khoảng cho 10-20 ứng dụng đó thực sự là một "thảm họa" nếu như mỗi ứng dụng bạn lại thích một **username** và **password** khác nhau (ý tôi là có thể hôm này bạn nghĩ ra **username** này hôm khác bạn thích **username** khác) hay là bạn phải lặp đi lặp lại công việc đăng ký **username** và **password**, rồi active qua email tới tận 10-20 lần.

Và nếu bạn là một tín đồ xem phim chẳng hạn, khi tham gia ứng dụng hoặc diễn đàn nào đó chắc hẳn các bạn đã gặp tình huống "Vui lòng đăng ký/đăng nhập thể xem link ẩn".

Bạn sẽ giải quyết thế nào trong tình huống tôi vừa đặt ra? Tôi tin là nhiều bạn sẽ có ngay câu trả lời là thôi mất chút thời gian đăng ký cũng được miễn là giải quyết được việc. Nhưng thật may mắn là chúng ta đã được cung cấp một giải pháp tuyệt vời hơn thế rất nhiều.

Nhận ra điều bất cập này, các "ông lớn" như Twitter, Facebook hay Google đã ngồi lại đàm phán với nhau và đưa ra một chuẩn mới có tên là **Open Authentication**. Trong nội dung của bài viết này, tôi sẽ giới thiệu với các bạn về chuẩn **Oauth** là gì? Cách thức hoạt động thế nào? Điểm mạnh và điểm yếu ra sao? Đồng thời phạm vi giới thiệu của tôi sẽ tập trung nhiều hơn vào chuẩn **Oauth** mới nhất hiện nay đó là Oauth2.

II Oauth2 và cơ chế hoạt động

1. Oauth là gì ?

Oauth là một phương thức chứng thực giúp các ứng dụng có thể chia sẻ tài nguyên với nhau mà không cần chia sẻ thông tin **username** và **password**. Từ **Auth** ở đây mang 2 nghĩa:

*.Authentication: xác thực người dùng thông qua đăng nhập.

*.Authorzation: cấp quyền truy cập.

Quay lại ví dụ ở phần mở đầu một chút, bạn có thể hiểu đơn giản là chúng ta đã đăng ký và có một tài khoản Facebook, và chúng ta sẽ dùng tài khoản này để đăng nhập ở 10-20 ứng dụng yêu thích của chúng ta.

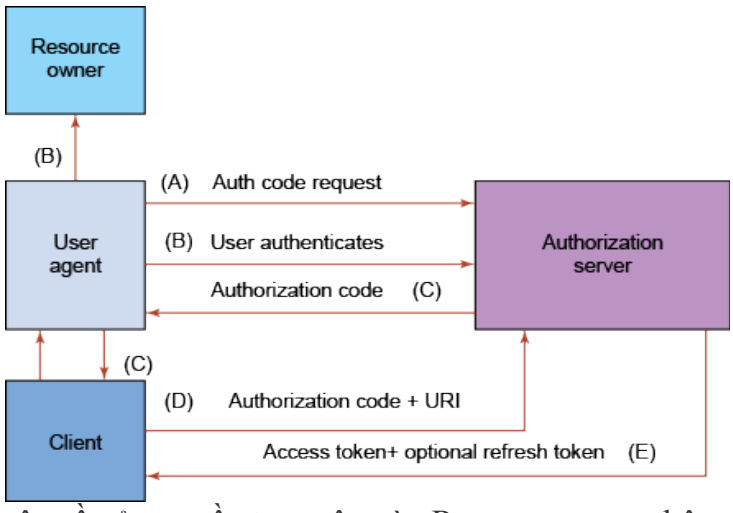
Tuy nhiên trường hợp không mong muốn xảy ra đó là một trong số 20 ứng dụng của chúng ta bị hack và hacker lấy được các thông tin người dùng của ứng dụng.Trong trường hợp này bạn cũng không phải lo lắng quá về việc sẽ mất tài khoản (username + password) Facebook vì ứng dụng bị mất của bạn chỉ được Facebook chia sẻ cho một chìa khóa (token) chứa quyền hạn nhất định và không được phép truy cập vào thông tin username cũng như password của bạn.

2. Các tác nhân trong Oauth2.

- Resource Owner(User):Là những người dùng ủy quyền cho ứng dụng cho phép truy cập tài khoản của họ. Sau đó ứng dụng được phép truy cập vào những dữ liệu người dùng nhưng bị giới hạn bởi những phạm vi (scope) được cấp phép. (VD: chỉ đọc hay được quyền ghi dữ liệu) -> đây là bạn.

- Client(Application):Là những ứng dụng mong muốn truy cập vào dữ liệu người dùng. Trước khi được phép tương tác với dữ liệu thì ứng dụng này phải qua bước ủy quyền của User, và phải được kiểm tra xác nhận thông qua API. ->đây là ứng dụng Garena Mobile chẳng hạn.
- Resourceserver(API):Nơi lưu giữ thông tin tài khoản User và được bảo mật.
- Authorization-Server :Làm nhiệm vụ kiểm tra thông tin User sau đó cấp quyền cho ứng dụng(Client) để truy cập cập Resourceserver thông qua access-token.

3. OAuth2 hoạt động như thế nào ?

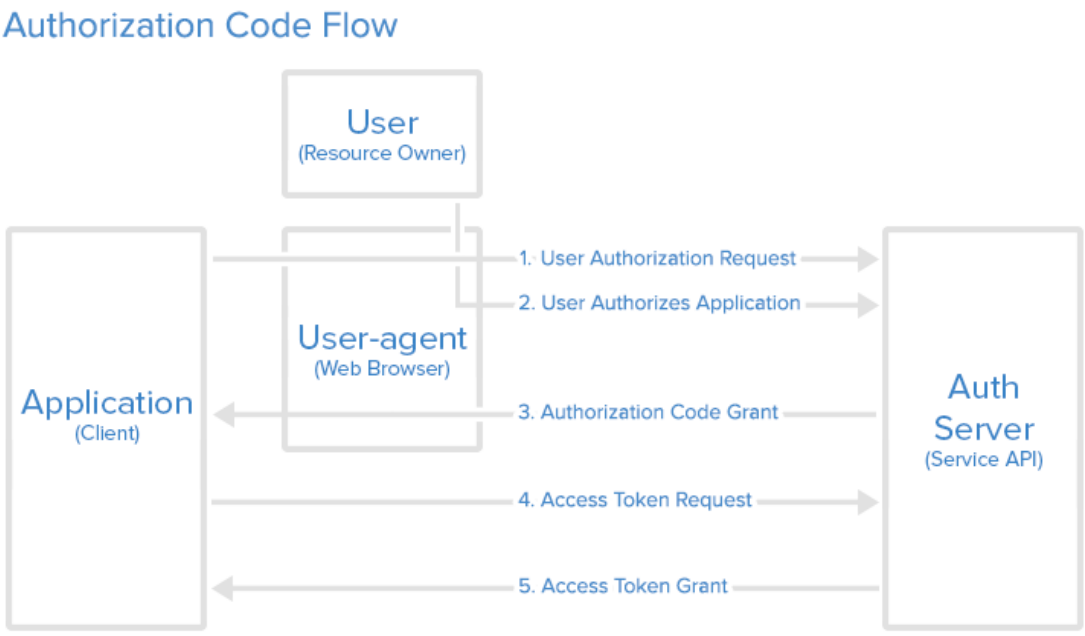


1. Application yêu cầu ủy quyền truy cập vào Resourceserver thông qua User.
2. Nêu User ủy quyền cho yêu cầu trên,App sẽ nhận được giấy ủy quyền của user (dưới dạng 1 token string nào đó chẳng hạn).
3. Application gửi thông tin định danh (ID) của mình kèm giấy ủy quyền từ phía user tới Authorzation server.
4. Nếu thông tin định danh được xác thực và giấy ủy quyền hợp lệ, Authozationserver sẽ trả về cho app 1 access-token.
5. Để truy cập tài nguyên (resources) từ Resourceserver và lây thông tin, Application sẽ phải đưa token để xác thực,
6. Nếu access-token hợp lệ, Resourceserver sẽ trả lại dữ liệu được yêu cầu cho application

III,Grant Type: Authorization Code & Grant Type: Resource Owner Password Credentials

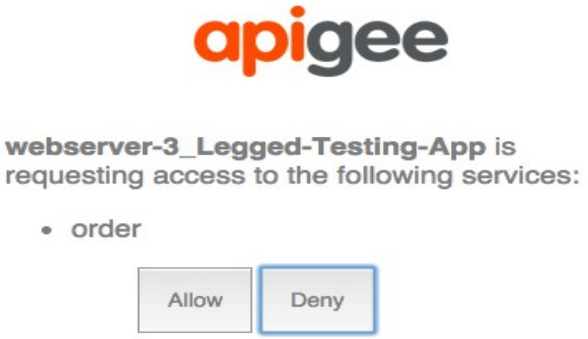
1.Grant Type:Authorization code

Đây là một hình thức ủy quyền được dùng phổ biến nhất hiện nay,sơ đồ hoạt động như sau:



1. Ứng dụng chuyển hướng người dụng lên Web-Browser(User-agent) tại đây người dùng xác nhận thông tin bản thân (đăng nhập username,password).

2. Người dùng xác nhận quyền mà application muốn sử dụng.



3. Sau khi thực hiện 2 bước trên web-browser thực hiện request lên Auth-Server để cấp code rồi trả về cho Client (Application).
4. Client gửi yêu cầu đổi code lấy token trực tiếp lên Auth-Server.
5. Server trả lại token cho Client.

2. Grant Type: Resource Owner Password Credentials

Với loại ủy quyền này, User sẽ phải cung cấp thông tin username và password trực tiếp cho Application sử dụng để lấy token. Cần lưu ý loại ủy quyền này chỉ nên sử dụng cho những ứng dụng thực sự được tin tưởng (VD: những ứng dụng của chính service hay những ứng dụng mặc định của hệ điều hành chẳng hạn). Flow ở đây hết sức đơn giản khi chỉ cần 1 request ta có thể lấy được token.

```
POST / http://author:8523/oauth20/tokens
Content-Type: application/x-www-form-urlencoded
grant_type=password&client_id=...&client_secret=....&username=...&password=.
...&scope=....
```

IV Sử dụng service Oauth2 theo flow Authorization code

1. Các thành phần trong service

a. Thành phần Auth-Server

Auth-Server bao gồm Author-Server và Authen-Server. Authen-Server làm nhiệm vụ xác nhận thông tin người dùng. Author-Server với xác nhận và cấp token, cung cấp tạo mới 1 application.Url của Author-Server và Authen-Server lần lượt là http://author:8523 và http://authen:9858 .

b, Thành phần User-agent (Web browser)

Cung cấp 1 trang 1 giao diện về đăng nhập (1 trang web) và hỏi quyền , 1 trang web về khai báo Application.

c, Thành phần Application (Client)

Trước khi sử dụng Oauth-server cho ứng dụng, bạn phải đăng ký ứng dụng với author-server một số thông tin cơ bản hay request như sau:

```
POST http://author:8523/oauth20/applications
Content-Type: application/json

{
"name": "testApp",
"description": "This is a test app",
"scope": "basic",
"redirect_uri": "http://10.46.16.93:8080",
"client_id": "b9db6d84dc98a895035e68f972e30503d3c724c8",
"client_secret":
"105ef93e7bb386da3a23c32e8563434fad005fd0a6a88315fcdf946aa761c838",
"application_details": {
"division": "IT",
"organization": "qm"
}
}
```

Trong đó:

- name:tên ứng dụng .
- description: description ứng dụng.
- scope: các phạm vi cho phép của ứng dụng.

- redirect_uri: chính là địa chỉ sẽ quay về sau khi quá trình ủy quyền kết thúc
- clientID và clientsecret: thông tin chứng thực của ứng dụng

*ClientID và ClientSecret nên là một chuỗi có dạng "qm"+System.currentTimeMillis

Chuỗi trả về có dạng:

```
Status Code: 200 OK
Content-Type: application/json
{
"client_id": "b9db6d84dc98a895035e68f972e30503d3c724c8",
"client_secret":
"105ef93e7bb386da3a23c32e8563434fad005fd0a6a88315fcdf946aa761c838"
}
```

Sau khi nhận được clientId và clientsecret tiếp theo bạn phải kích hoạt ứng dụng bằng 1 request có dạng:

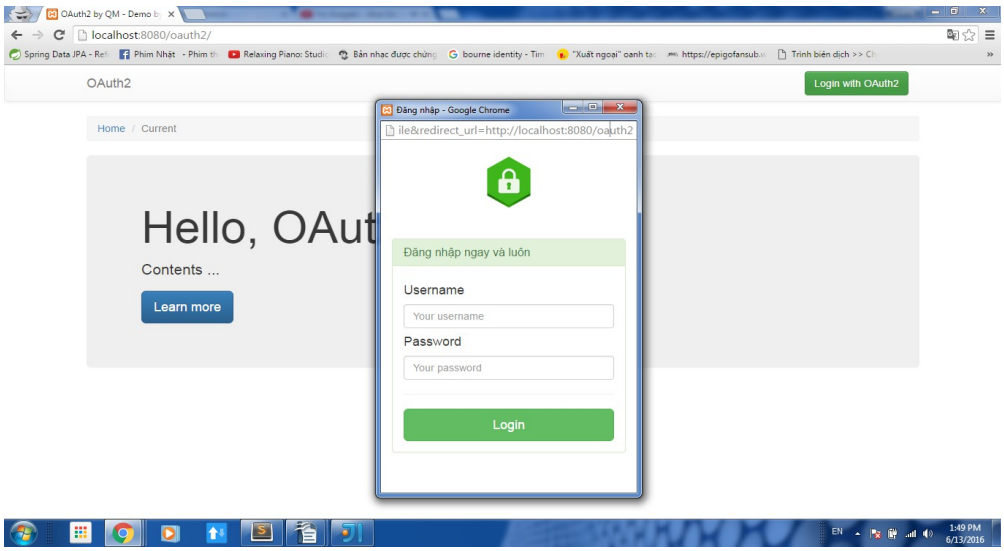
```
PUT
http://author:8523/oauth20/applications/b9db6d84dc98a895035e68f972e305
Content-Type: application/json
{"status":1}
```

d, Thành phần Resourceserver

1 Web-Service chứa tài nguyên của người sử dụng (API). Để truy cập vào Resourceserver cần 1 khóa (token) có chứa các phạm vi đã được quy định của Application.

2. Xây dựng 1 ứng dụng sử dụng Oauth2

Đầu tiên người dùng được chuyển hướng lên trang xác thực người dùng và xác thực quyền của application tại trang web của OauthServer.



1 đường request ngầm sẽ thực hiện đồng thời việc xác thực và xin auth-code vd:

GET http://authen:9858/getCode?username=...&password=...&client_id=...&redirect_uri=...&scope=....

Trong đó

- authen:9858 thuộc authen-server
- client_id:ClientId của application sau khi đăng ký với Service
- redirect_uri:Nơi quay về sau khi xác thực xong và sau khi sinh ra auth-code
- scope:định nghĩa phạm vi quyền cho phép Application truy cập(vd:read_profile,read_list_friend)

Với response thành công .

Status code: 200
"redirect_uri?code=....."

Với response thất bại.

Status code: 400
"invalid auth_code"

Sau khi nhận được auth-code, Application cần thực hiện Request lên OauthServer lên một lần nữa lấy access-token.Việc này được Application đảm nhận và user không cần thao tác thêm gì nữa.Link request có dạng

POST http://author:8523/oauth20/tokens
Content-Type:application/x-www-form-urlencoded
grant_type=authorization_code &code =...&redirect_uri =...&client_id =...&client_secret=...

với response thành công .

```
Status Code: 200 OK
Content-Type: application/json
{
"access_token":
"857fd034dabea45bd130ef3f4af4d7929118d91974e00f399335794889404349",
"refresh_token":
"b8d4a3bcecdb53479cf59bcb724840123d795a7a3e6b7af6ea43d532436ff44d",
"token_type": "Bearer",
"expires_in": "120",
"scope": "basic",
"refresh_expires_in": "300"
}
```

(thời gian hết hạn expires_in được chiếu theo thời gian quy định của scope.Vd:scope: "read_profile" là 360000s thì access_token cũng như vậy)

với response không thành công

```
Status Code: 400 Bad Request
Content-Type: application/json
{"error": "invalid auth_code"}
```

Tới đây Application đã được User ủy quyền truy cập.Application có thể sử dụng access_token để truy cập những tài nguyên (Resourceserver) của dịch vụ mà User cho phép, ví dụ thông tin , ảnh avatar, ... cho tới khi access_token hết hạn sử dụng. Nếu phía API hỗ trợ và gửi về thêm cả thông tin refesh_token thì Application thể sử dụng để đổi lấy access_token mới khi access_token cũ hết hạn.

3, Thư viện FilterEx trong cho Resourceserver

Thư việc FilterEx được viết ra để hỗ trợ cho việc quản lý token. Kiểm tra token hợp lệ hay không.

Sau khi application giữ được token. Để truy cập tài nguyên(API) Resourceserver, với mỗi request gửi lên server cần gắn token vào header trước vd:

'token':"1fa18309dade07a9e6e02ce0a66c0d5f04653309f9a20eec6b4956748d640e91"

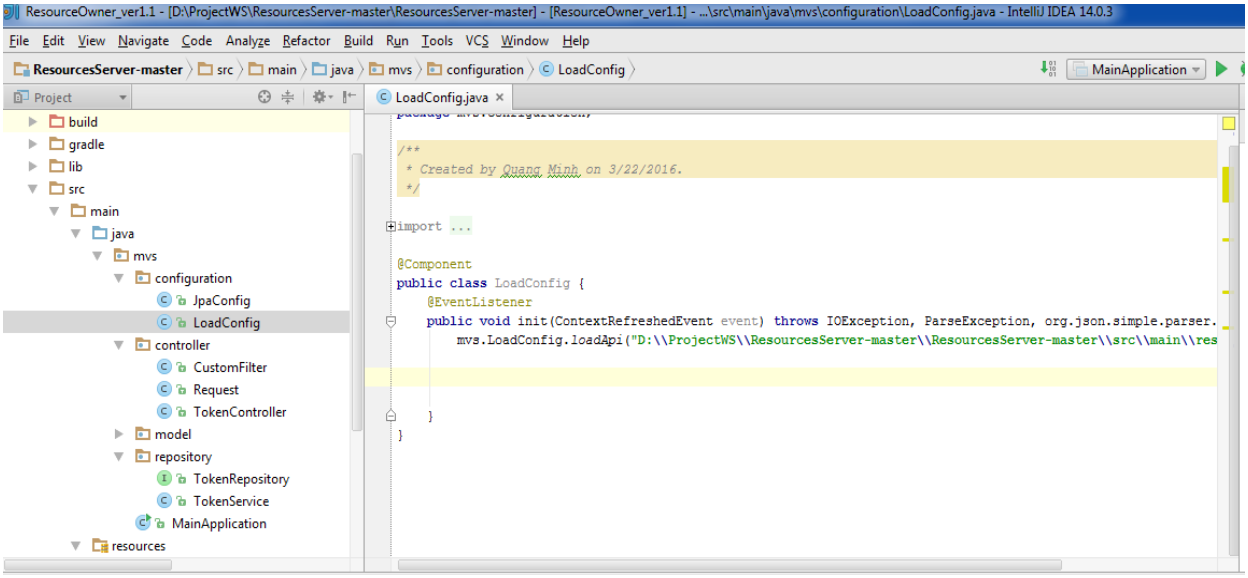
Để sử dụng thư viện ta cần cấu hình 1 file config.json.

```
{
  "url_server":"http://author:8523",
  "api":[
    {"url":"/tokens/testapi1","scopes":["read_profile"],"method":"GET"},
    {"url":"/tokens/testapi2","scopes":
["read_profile","read_list_friend"],"method":"GET"},
    {"url":"/account/{accountId}/friend/{friendId}","scopes":
["read_profile"],"method":"GET"},
    {"url":"/tokens","scopes":[],"method":"GET"}
  ]
}
```

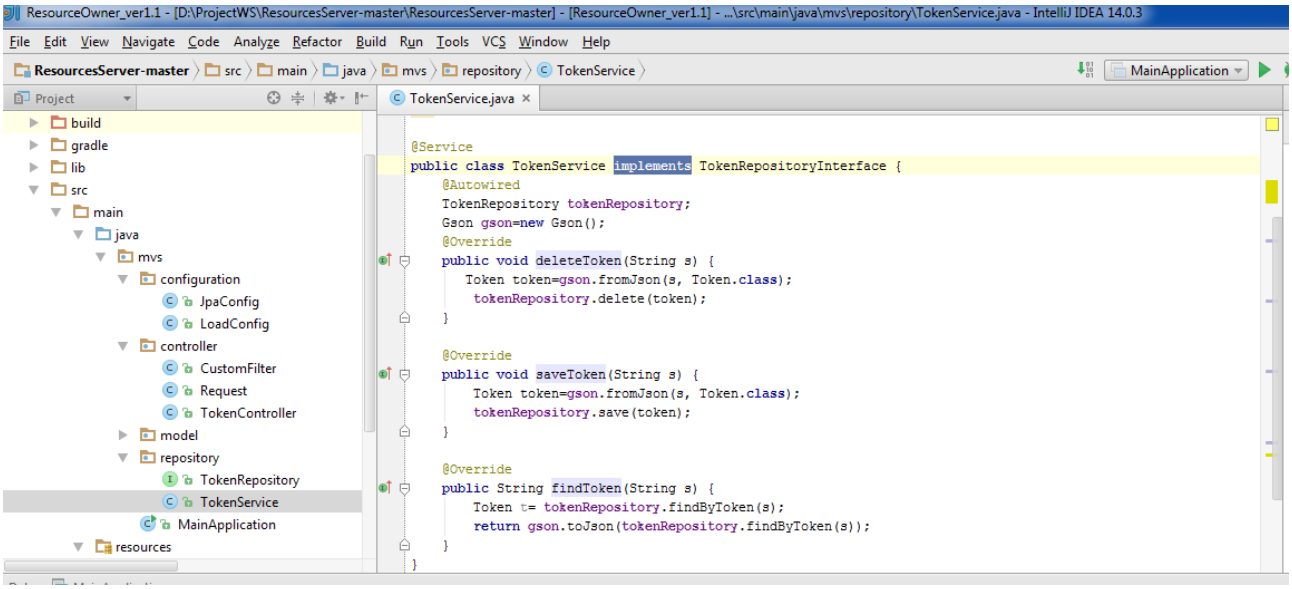
Trong đó:

- url_server: đường dẫn đến author-server.
- Api thuộc về resourceserver bao gồm uri, scopes, method

Và việc đọc file được thực hiện trước khi sử dụng thư viện FilterEx với hàm mvs.LoadConfig.LoadApi(FileReader filereader) hoặc đầu vào là đường dẫn file config.json.



Thư viện FilterEx hỗ trợ việc lưu lại token tại chính Resourceserver nhằm tăng tốc độ tránh việc liên tục gửi request lên author-server. Để sử dụng cần phải implement interface "TokenRepositoryInterface" .VD class TokenService implement TokenRepositoryInterface



Việc sử dụng thư viện FilterEx được sử dụng trong lớp CustomFilter. Response Oauth2Filter.checkToken(TokenRepositoryInterface tokenService,Servlet request) (trong đó param 1 có thể null)