**BẢN CHẤT QUY TRÌNH ĐĂNG NHẬP VỚI GOOGLE VỚI OAuth( open authorization):**

1. **Người dùng** muốn sử dụng ứng dụng bên thứ ba (client app).
2. **Client app** chuyển hướng người dùng đến dịch vụ (như Google).
3. **Dịch vụ** thực hiện xác thực người dùng.
4. **Nếu xác thực thành công**, dịch vụ hỏi người dùng xem họ có muốn cấp quyền cho **client app** truy cập vào thông tin của họ hay không.
5. **Nếu người dùng đồng ý**, dịch vụ cấp cho **client app** một **Access Token qua giao thức OAuth**.
6. **Client app** sử dụng **Access Token** để truy cập vào thông tin của người dùng từ dịch vụ.

LƯU Ý:

- Khi Google đã xác thực người dùng nhưng **chưa cấp quyền**, thì **client app** (ứng dụng bên thứ ba) chưa hoàn thành quá trình xác thực người dùng !

*=> app client xác thực người dùng = xác thực của Google + cấp quyền qua giao thức OAuth*

**Credentials**  là các thông tin mà ứng dụng của bạn sử dụng để xác định danh tính khi yêu cầu quyền truy cập đến một dịch vụ hoặc API của Google. Để truy cập vào các tài nguyên của người dùng (như email, lịch, tệp trong Google Drive, v.v.), bạn cần cung cấp thông tin xác thực bao gồm Client ID và Client Secret.

Các thành phần trong Credentials:

• Client ID

• Client Secret

• Redirect URI

• Authorization Code

• Access Token

**Tóm tắt quy trình sử dụng các thành phần:**

1. Ứng dụng của bạn yêu cầu người dùng đăng nhập và cấp quyền bằng cách chuyển hướng người dùng tới Google Authorization Server với Client ID và Redirect URI.

2. Người dùng đăng nhập và chấp nhận quyền truy cập.

3. Google Authorization Server chuyển hướng người dùng về Redirect URI của bạn cùng với Authorization Code.

4. Ứng dụng của bạn sử dụng Authorization Code, Client ID, và Client Secret để gửi yêu cầu tới Google Token Endpoint và đổi lấy Access Token.

5. Với Access Token, ứng dụng của bạn có thể truy cập tài nguyên của người dùng qua Google Resource Server (ví dụ: Google Drive API).

public static String getToken(String code) throws ClientProtocolException, IOException {

// call api to get token

String response = Request.Post(GlobalConfig.GOOGLE\_LINK\_GET\_TOKEN)

.bodyForm(Form.form().add("client\_id", GlobalConfig.GOOGLE\_CLIENT\_ID)

.add("client\_secret", GlobalConfig.GOOGLE\_CLIENT\_SECRET)

.add("redirect\_uri", GlobalConfig.GOOGLE\_REDIRECT\_URI).add("code", code)

.add("grant\_type", GlobalConfig.GOOGLE\_GRANT\_TYPE).build())

.execute().returnContent().asString();

JsonObject jobj = new Gson().fromJson(response, JsonObject.class);

String accessToken = jobj.get("access\_token").toString().replaceAll("\"", "");

return accessToken;

}

Đoạn mã này có nhiệm vụ gửi một yêu cầu POST đến API của Google để đổi **authorization code** (mã ủy quyền) thành **access token** trong quy trình OAuth 2.0. Sau đó, nó sẽ xử lý phản hồi từ API và trả về **access token** dưới dạng chuỗi. Cụ thể cú pháp được giải thích như sau:

**1. Request.Post(GlobalConfig.GOOGLE\_LINK\_GET\_TOKEN):**

* Request.Post() là một phương thức của thư viện **Apache HttpClient Fluent API** được sử dụng để thực hiện một yêu cầu HTTP POST.
* GlobalConfig.GOOGLE\_LINK\_GET\_TOKEN chứa URL của Google API, nơi ứng dụng sẽ gửi yêu cầu POST để lấy **access token**.

**2. .bodyForm(Form.form().add(...)...build()):**

* bodyForm() được sử dụng để gửi các thông tin dưới dạng một **form data** (dữ liệu biểu mẫu).
* Form.form() là đối tượng đại diện cho dữ liệu biểu mẫu của request. Các cặp key-value được thêm vào biểu mẫu này bằng các phương thức .add().
* Các tham số cần thiết để lấy **access token** từ Google được truyền vào biểu mẫu này:
  + **client\_id**: ID của ứng dụng client được Google cung cấp khi bạn đăng ký OAuth.
  + **client\_secret**: Chuỗi bảo mật của ứng dụng client, dùng để xác minh ứng dụng với Google.
  + **redirect\_uri**: Địa chỉ mà Google sẽ chuyển hướng sau khi xác thực xong (URL phải khớp với URL đăng ký với Google).
  + **code**: Authorization code mà Google trả về trước đó trong bước xác thực.
  + **grant\_type**: Loại yêu cầu trong OAuth. Ở đây là authorization\_code, nghĩa là ứng dụng đang dùng mã ủy quyền để lấy access token.

Cuối cùng .build() sẽ xây dựng đối tượng Form để gửi đi trong body của yêu cầu POST.

**3. .execute():**

* Phương thức này thực hiện yêu cầu POST đến Google API.

**4. .returnContent().asString():**

* Sau khi yêu cầu được thực hiện thành công, nội dung phản hồi từ Google API sẽ được trả về dưới dạng chuỗi và gán vào biến response.
* Google sẽ trả về một JSON chứa thông tin như access token, loại token, thời gian hết hạn token, v.v.

**5. new Gson().fromJson(response, JsonObject.class):**

* **Gson** là thư viện Java được sử dụng để chuyển đổi giữa JSON và các đối tượng Java.
* response (chuỗi JSON trả về từ Google) sẽ được chuyển thành một đối tượng **JsonObject** bằng cách sử dụng phương thức fromJson() của Gson.

**6. String accessToken = jobj.get("access\_token").toString().replaceAll("\"", "");:**

* Sau khi nhận được đối tượng JSON, cú pháp này sẽ lấy giá trị của trường "access\_token" từ JSON.
* .replaceAll("\"", "") loại bỏ các dấu ngoặc kép dư thừa xung quanh giá trị của token.

**7. return accessToken;:**

* Cuối cùng, **access token** được trả về dưới dạng chuỗi để sử dụng cho các yêu cầu tiếp theo.

Access token

{

"access\_token": "ya29.A0ARrdaM9w-sample-access-token",

"expires\_in": 3599,

"scope": "email profile",

"token\_type": "Bearer",

"refresh\_token": "1//0gO-sample-refresh-token"

}

userDto

{

"id": "123456789",

"email": "user@example.com",

"verified\_email": true,

"name": "John Doe",

"given\_name": "John",

"family\_name": "Doe",

"picture": "https://lh3.googleusercontent.com/a-/AOh14Gh-sample-photo-url",

"locale": "en"

}