Google OAuth 2.0

Cập nhật: tháng 5/2019

Contents

1. Vắn tắt về dịch vụ Google OAuth

1.1. Yêu cầu thực tế

- Mô hình khởi điểm: user cần xác thực (bằng phương thức login) trước khi có thể truy nhập đến thông tin của mình được lưu trên server.
- Mỗi user tham gia nhiều dịch vụ ==> nhiều server xác thực ==> nhiều tài khoản xác thực ==> khó khăn cho user
- Giải pháp single-sign-on giúp nhiều dịch vụ dùng chung một tài khoản xác thực cho mỗi user (các dịch vụ thực thi cơ chế login riêng nhưng với cùng một tài khoản và mật khẩu). Giải pháp được biết đến nhiều là LDAP (Lightweight Directory Access Protocol), tuy nhiên tích hợp giải pháp này trong việc xây dựng ứng dụng khá phức tạp.
- Email (mà cụ thể là Gmail) thực tế đã trở thành ID của mỗi người. Ai cũng có tối thiểu một email để trao đổi thông tin trên Internet. Ai sử dụng điện thoại Android cũng có một tài khoản Gmail. Các tài khoản Gmail này đã không còn chỉ dùng để trao đổi thông tin mà dần dần là tài khoản xác thực của user.
- Cần mô hình chuẩn để chia sẻ tài khoản xác thực user (Gmail) cũng như cho phép các dịch vụ cộng tác với nhau trong quá trình xác thực user (authentication) và cấp quyền truy nhập (authorization) đến các tài nguyên của mỗi dịch vụ ==> ra đời Chuẩn OAuth[1] và OpenID[2]

1.1. Google OAuth

OAuth là dịch vụ Google cung cấp, cài đặt theo chuẩn OAuth 2.0 và OpenID, cho phép thực thi các nhiệm vụ liên quan đến xác thực (authentication) và cung cấp quyền truy nhập (authorization), cụ thể như sau:

- Xác thực user bằng tài khoản Google trên một ứng dụng Web bất kỳ
- Cho phép truy nhập đến các tài nguyên Google của user (như là email, calendar, v.v…) mà không cần biết mật khẩu truy nhập tài khoản của user
- Lưu thông tin sử dụng dịch vụ (charging) để thực hiện cơ chế thu phí (billing).
- ???

1.1.1. Các bước sử dụng

Trực quan từ phía người sử dụng, các bước thực hiện với Google OAuth có thể tóm tắt như sau:

- Người sử dụng yêu cầu truy nhập vào một ứng dụng nào đó của Google hoặc của một nhà cung cấp dịch vụ khác. Ứng dụng yêu cầu xác thực user.
- Người sử dụng được điều hướng đến trang Web login của Google với lời nhắc hãy chọn tài khoản Google của mình và login.

1 https://en.wikipedia.org/wiki/OAuth
2 https://en.wikipedia.org/wiki/OpenID

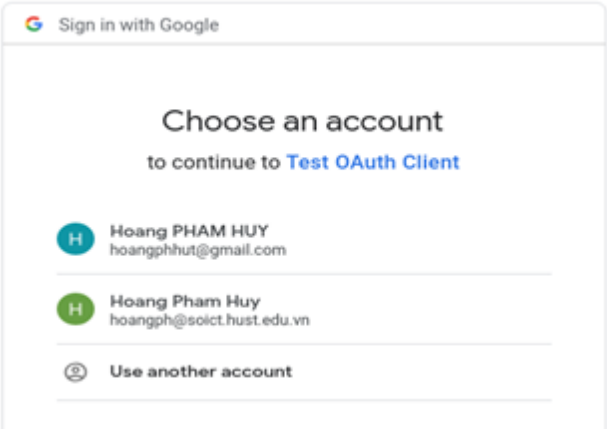


Figure 1: Xác thực với Google account để làm việc với App "Test OAuth Client"

- Trong một số trường hợp, người sử dụng được Google nhắc nhở có cho phép ứng dụng được truy nhập đến các tài nguyên của user hay không:

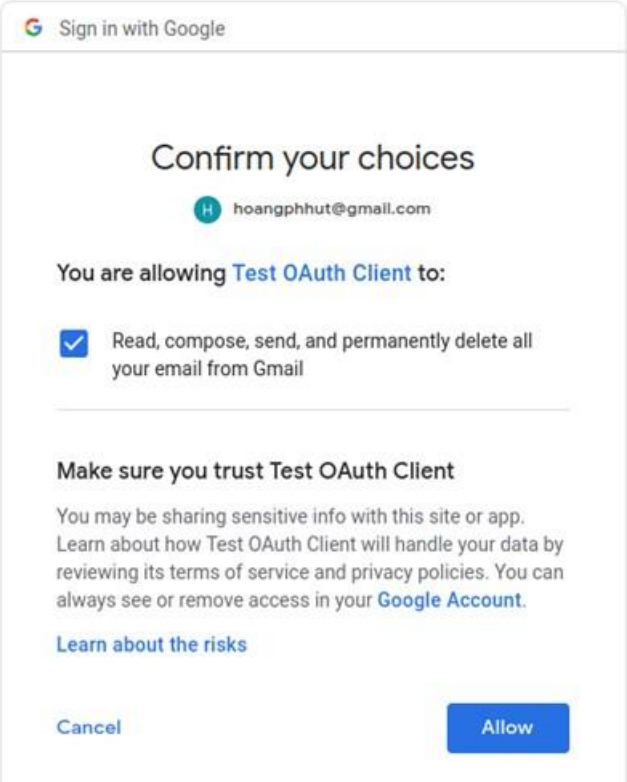


Figure 2: User xác nhận cho phép App truy nhập đến dữ liệu của mình

- Sau khi login và (nếu có yêu cầu) cho phép một số quyền truy nhập đến các dữ liệu/tài nguyên của mình, người sử dụng được xác nhận bằng tài khoản Google và bước vào phiên làm việc với ứng dụng. Phiên làm việc sẽ kết thúc khi hết thời gian được phép, hoặc khi người sử dụng logout khỏi phiên.

1.1.2. Mô hình kết nối

Để triển khai các bước sử dụng như trên, nhiều thành phần của ứng dụng được kết nối với các dịch vụ có liên quan của Google theo mô hình OAuth (phiên bản 2.0).

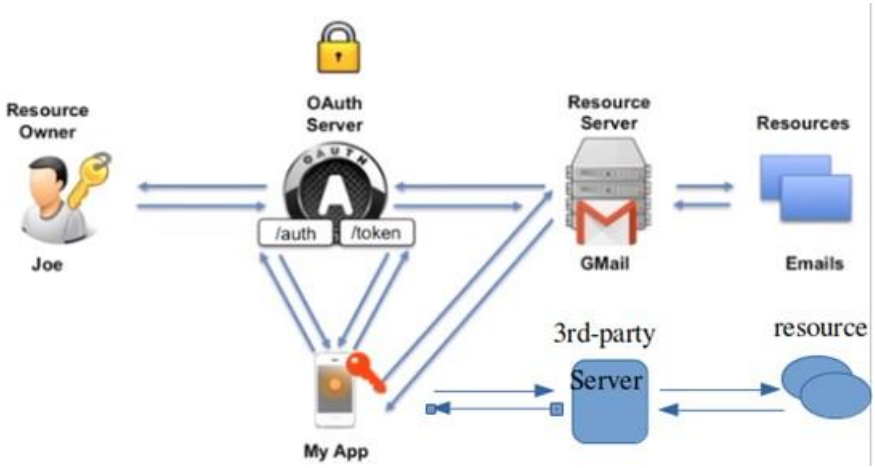


Figure 3: Mô hình tích hợp các thành phần Google OAuth

Các khái niệm:

- Resource owner: là người sở hữu mail (trường hợp này là người sử dụng có email hoangph@soict.hust.edu.vn)
- OAuth Server: là hệ thống Google Auth (được xác định bằng URI https://accounts.google.com)
- App: là ứng dụng của bên thứ 3 muốn truy nhập vào resource (với sự cho phép của resource owner). App có thể là một ứng dụng Web, Mobile hoặc trên PC.
- Resource server: là nơi quản lý resource. Nó có thể là một server của Goole hoặc một server của bên thứ 3. App sẽ truy nhập đến resource bằng API mà Resource server cung cấp.

Các bước tích hợp giữa các thành phần:

1. Đây là bước chuẩn bị và không có trong sơ đồ bên trên. App đăng ký với Google Auth server để có mã số định danh (Client ID) và mã bí mật (secret code). Mã bí mật này có vai trò như "mật khẩu" của App cần được cấp cho Google Auth server cùng với Client ID khi liên lạc để xác thực App. Bước đăng ký App với Google Auth server đồng thười cũng xác định luôn các chức năng (Google API) mà App muốn sử dụng. Những chức năng này sẽ được cung cấp cho user (dưới cách điễn giải dễ hiều) trong bước Google Auth server xác thực user và cấp quyền App được truy nhập đến các tài nguyên của user.

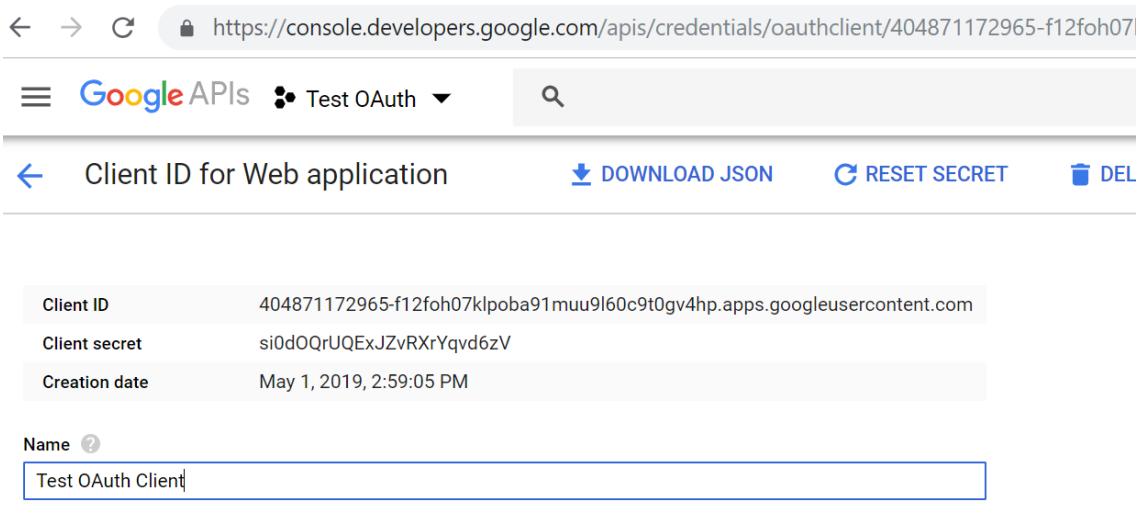


Figure 4: App đăng ký Client ID & secret code với Google Auth server

2. User bắt đầu truy nhập vào App và cần phải được xác thực. App điều hướng user đến trang login chuẩn của Google để yêu cầu xác thực user bằng tài khoản Google và password. Nếu App là một ứng dụng Web, thường có một button "Google Sign-in" để thực hiện điều hướng.

   Cú pháp url điều hướng đến trang Google login tuân thủ theo chuẩn OAuth 2.0 (RFC 6749, mục 4.1.1: Authorization Request[1]). Khi chuyển sang url link điều hướng có dạng sau:

```
https://accounts.google.com/o/oauth2/auth?
        response_type=code&
        client_id=<Client ID>&
        redirect_uri=<Redirect URI>&
        scope=https%3A%2F%2Fmail.google.com&
        approval_prompt=force
```

   Trao đổi về vấn đề bảo mật. Trong bước này, user phải cũng cấp login email và mật khẩu Google nên tiềm ẩn nhiều nguy cơ bảo mật. Ví dụ, một ứng dụng Web hay App có thể điều hướng giả mạo đến trang login không phải của và user đăng nhập sẽ bị lộ thông tin tài khoản. Trường hợp nguy hiểm hơn là khi dịch vụ DNS bị xâm nhập, thay đổi dữ liệu làm cho ứng dụng App điều hướng đến trang login Google nhung bị phân giải IP sang server giả mạo. Việc xác định trang login là thật hay giả hoàn toàn dựa vào khả năng của user và đây là một điểm yếu của mô hình hiện tại. Ngoài khả năng nhận dạng trang Google login "thật" theo giao diện (hầu như không hiệu quả) thì user có thể nhận dạng theo địa chỉ trang (khó giả mạo hơn, nhung không phải là không thể).

3. Bước này chỉ xuất hiện khi App muốn truy nhập đến tài nguyên Googe của user (trong mô hình Google Sign-in cài đặt theo OpenID sẽ không có bước này). User login thành công ở trang web Google Auth server sẽ nhận được thông báo đề nghị cho phép App truy nhập vào resource. Thông báo này chính là diễn giải dễ hiểu theo nguồn ngữ người sử dụng về các Google API mà App đã đăng ký sử dụng trong bước đầu tiên.

   "Sự cho phép" của user này được thể hiện bằng một mã code do Google Auth tạo ra khi user click button Allow và cần được chuyển cho App để sử dụng trong bước sau. Phương pháp để Google Auth server chuyển mã code cho App là sử dụng redirect_uri

1 https://tools.ietf.org/html/rfc6749#section-4.1.1

(đã được App chuyển cho Google Auth server trong bước 2). App cần sẵn sàng ở địa chỉ redirect_uri này để nhận giá trị mã code (cùng một số thông tin khác) và xử lý bước tiếp theo. Cú pháp kết quả trả về trong bước này được mô tả trong chuẩn OAuth 2.0 (RFC 6749, mục 4.1.2: Authorization Response[1]). Ví dụ nếu redirect_uri có giá trị là http://localhost thì Google Auth server sẽ gửi mã code theo đường link như sau:

```
http://localhost/?
        code=4/PQFzt57ZVE1FkWGeR_VMQrxvMAMvr8WKx2e1kVHaOIMkWOcMQzXFgNuBEPrPIkmpPy1Df-
4Svp-hLiHGmOhHYNs&
        scope=https://mail.google.com/
```

Thảo luận về mã code. Có thể diễn giải ý nghĩa của mã code này như một "thẻ thông hành" mà chủ sở hữu resource chứng nhận cho phép App được truy nhập vào resource. Tuy nhiên, không có gì đảm bảo người cầm mã code này chính là App (nên cần có bước App xác thực với Google bằng secret code). Một điểm nữa là mã code được gửi từ Googe Auth đến App qua redirect_uri ở dạng public, nên mã code này hoàn toàn không có tính bảo mật.

4. App cung cấp mã code cho Google Auth server để nhận access token. Trong bước này, App cũng cần xác thực với Google Auth server bằng Client ID và mã bí mật của mình (đã đăng ký với Google Auth server trong bước đầu tiên). Mặc định access token này có hiệu lực trong 3600s và cần phải được đính kèm lời gọi API mỗi khi App muốn truy nhập đến tài nguyên của Google. Cú pháp API yêu cầu access token tuân thủ theo chuẩn OAuth 2.0 (RFC 6749, mục 4.1.3: Access Token Request[2]). Ví dụ sau mô tả cú pháp sử dụng *curl* để gọi đến API này:

```
curl -X POST \
        -H "content-type: application/x-www-form-urlencoded" \
        -d "grant_type=authorization_code&code=<Code>& \
                redirect_uri=<Redirect URI>& \
                client_id=<Client ID>& \
                client_secret=<Secret Code>" \
        "https://accounts.google.com/o/oauth2/token"
```

Kết quả trả về theo cú pháp chuẩn OAuth 2.0 (RFC 6749, mục 4.1.4: Access Token Response[3]). Ví dụ:


```
{
  "access_token":
"ya29.Glv8Bq4uWFgeM2h5aF1Y_LpNfc4XHpmIOTzhYm_6jmUKnuRoOiKPQambJnBKIj8wp4dmmyckFXDsSLh64
QaTQ7Kd9h1u0t6FSNkeFrBDhJPK-amK84ai3zRb639Y",
  "expires_in": 3600,
  "scope": "https://mail.google.com/",
  "token_type": "Bearer"
}
```


Thảo luận về mức độ bảo mật của access token. Access token an toàn hơn rất nhiều so với code. Lý do như sau:

(a) code được gửi từ trang web login của Goole (vốn là trang web public) đến App (tại địa chỉ redirect_uri). Như vậy hoàn toàn có thể bị nghe lén, hoặc giả mạo. Kết quả là người giữ code này chưa chắc đã phải là App.

(b) access token được Google Auth server tạo ra khi App cung cấp Client ID và Secret code (là giá trị chỉ có App và Google Auth server biết). Hơn nữa, liên lạc

1 https://tools.ietf.org/html/rfc6749#section-4.1.2
2 https://tools.ietf.org/html/rfc6749#section-4.1.3
3 https://tools.ietf.org/html/rfc6749#section-4.1.4

giữa App và Google Auth server là liên lạc "private" (khác với liên lạc giữa user và Google Auth server được public ở trang Sign-in).

Vì độ bảo mật này nên Google xử lý "khắt khe" với code (ngoài việc có thời hạn sử dụng, code được dùng 1 lần, khi đã tạo access token thì code hết giá trị).

5. App gắn access token vào header của mỗi HTTP request khi gửi đến Google Resource server để yêu cầu truy nhập đến Google resource.

   Ở đây có một bước liên lạc "không tường minh" (implicit) giữa Google Auth server với Resource server để xác lập access token (khi Google Auth server cung cấp access token thì Resource server cũng phải hiểu ý nghĩa của access token này để xử lý phù hợp khi App gọi đến). Trường hợp cả 2 server đều là của Google (như ví dụ demo App truy nhập Gmail bên dưới) thì liên lạc giữa 2 server là một hộp đen (nên gọi là "không tường minh"). Trường hợp Resource server là của bên thứ 3 (ví dụ trong demo Google Sign-in) thì mô hình này không áp dụng được (không có liên lạc giữa Google Auth server và Resource server). Thay vào đó, mỗi Google account phải được ánh xạ vào một tài khoản của Resource server và mô hình Google Account Sign-in được áp dụng (Google Sign-in cài đặt theo chuẩn OpenID).

2. Tìm hiểu hệ thống Google OAuth bằng các ứng dụng demo

2.1. Chuẩn bị môi trường

1. Cài đặt *curl* để thực hiện các truy vấn HTTP: *sudo apt-get install curl*
2. Eclipse for Java EE: https://www.eclipse.org/downloads/packages/
3. Tạo Java project:
   - Kiểu project: Web / Dynamic Web
   - Target runtime: Chọn Apache Tomcat 7. Nếu chưa có thì chọn New runtime Apache Tomcat 7.0 & click <Download and Install...>

2.2. Xây dựng ứng dụng truy nhập Gmail

Thực hiện các bước theo Youtube "How Google is using Oauth".

Ứng dụng Web được triển khai bằng tool *curl*.

2.2.1. Bước 1: Đăng ký App với Google

- https://console.developers.google.com
- Tạo project: Test OAuth
- Chọn menu Library, seach Gmail, chọn Gmail API & thiết lập "Enable". Bước này đăng ký project "Test OAuth" được gọi Gmail API.
- Chọn menu Credential, yêu cầu "Create credentials" và chọn loại "OAuth client ID". Application type: Web application (chú ý chọn đúng kiểu, chọn sai kiểu bước lấy access token sẽ không chạy), Name: Test OAuth Client. Authorized redirect URIs =

http://localhost (đây là các URI được cho phép sử dụng như redirect_uri sau này). Kết quả:

Here is your client ID:

404871172965-f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com

Here is your client secret:

si0dOQrUQExJZvRXrYqvd6zV

Chú ý: khi tạo credential, cần đưa vào các thông số cho "OAuth consent screen". Đây là các thông tin sẽ hiển thị trên màn hình Google login để thông báo cho user về ứng dụng đang yêu cầu được authenticate. Tạm thời chỉ đưa vào duy nhất thông tin Application name: Test OAuth Client.

2.2.2. Bước 2: Xác thực với Google

Flow như sau:

- user được redirect đến trang Google login để chọn account sẽ thực hiện cho ứng dụng đã đăng ký (Test OAuth Client)

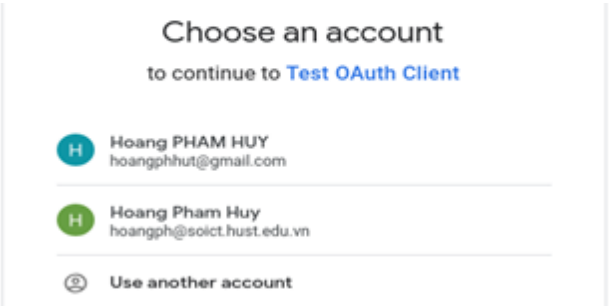


Figure 5: Xác thực với Google account để làm việc với App "Test OAuth Client"

- Sau khi user chọn account và login, Google Authen System sẽ hiển thị thông tin chi tiết về quyền mà ứng dụng Test OAuth Client đang muốn thực hiện với Google account vừa chọn:

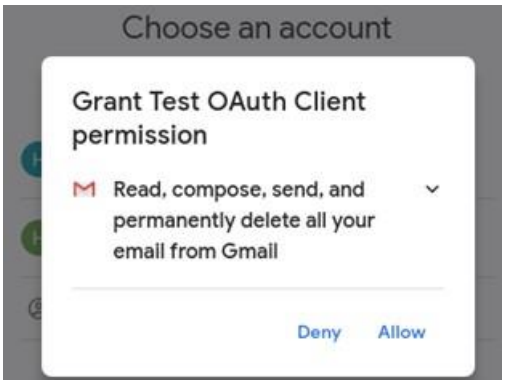


Figure 6: Cho phép (grant) quyền truy nhập tài nguyên

- Vì là bước quan trọng, Google sẽ xác nhận lại thêm lần nữa:

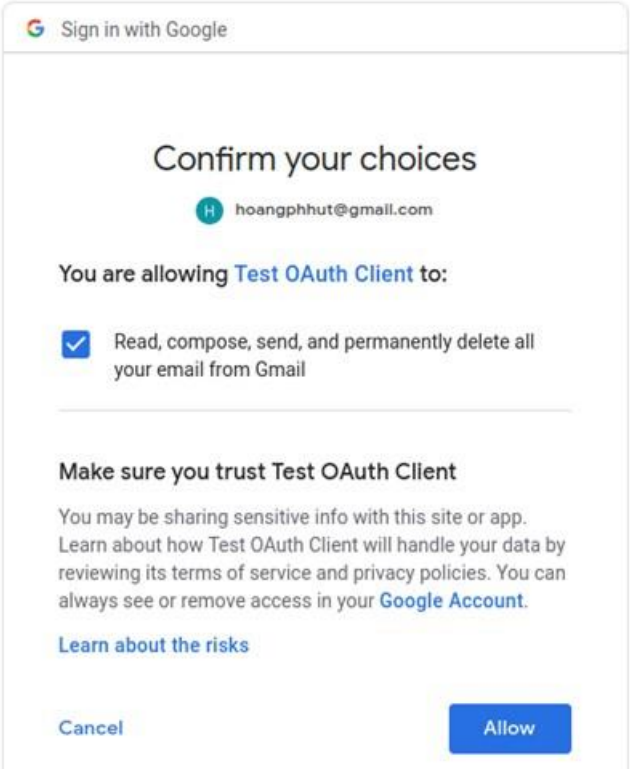


Figure 7: User xác nhận cho phép App truy nhập đến dữ liệu của mình

- User trả lời đồng ý (Allow) hoặc không (Cancel), Google đều redirect đến URI đã được cũng cấp cho Google. Trường hợp Cancel, redirect với tham số error=access_denided:

  http://localhost/?error=access_denied

  Trường hợp đồng ý, Google redirect với tham số code và scope trả về mã code sẽ được sử dụng để truy nhập đến resource (Gmail) của user trong bước sau:

  ```
  http://localhost/?code=4/PQFzt57ZVE1FkWGeR_VMQrxvMAMvr8WKx2e1kVHaOIMkWOcMQzXFgNuBEPrPIk
  mpPy1Df-4Svp-hLiHGmOhHYNs&scope=https://mail.google.com/
  ```

  Như vậy, ứng dụng cần được cài đặt và đợi ở redirect URI này và nhận các giá trị code & scope.

API để thực hiện authenticate Google như sau:

```
https://accounts.google.com/o/oauth2/auth?redirect_uri=http%3A%2F%2Flocalhost&response_t
ype=code&client_id=404871172965-
f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com&scope=https%3A%2F%2Fmail.google.c
om&approval_prompt=force
```

Trong API này phải đưa các thông tin client ID và redirect URI để Google xử lý như các bước nêu trong flow. Các URI này cần được mã hóa (encode) để thay thế các ký tự đặc biệt. Có nhiều tool mã hóa URI trên mạng, ví dụ: https://www.urlencoder.org/

2.2.3. Bước 3: Đổi code để lấy access token

Ứng dụng sau khi nhận được code cần yêu cầu Google Auth server cung cấp access token. Để đơn giản, demo sử dụng *curl* để tạo HTTP request đến Google cùng với các thông tin cần thiết để nhận access token:

```
curl -X POST -H "content-type: application/x-www-form-urlencoded" \
       -d "grant_type=authorization_code& \
       code=4/PQFzt57ZVE1FkWGeR_VMQrxvMAMvr8WKx2e1kVHaOIMkWOcMQzXFgNuBEPrPIkmpPy1Df-
4Svp-hLiHGmOhHYNs& \
       redirect_uri=http%3A%2F%2Flocalhost& \
       client_id=404871172965-
f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com& \
       client_secret=si0dOQrUQExJZvRXrYqvd6zV" \
       "https://accounts.google.com/o/oauth2/token"
```

Kết quả trả về:


```
{
  "access_token":
"ya29.Glv8Bq4uWFgeM2h5aF1Y_LpNfc4XHpmIOTzhYm_6jmUKnuRoOiKPQambJnBKIj8wp4dmmyckFXDsSLh64Q
aTQ7Kd9h1u0t6FSNkeFrBDhJPK-amK84ai3zRb639Y",
  "expires_in": 3600,
  "scope": "https://mail.google.com/",
  "token_type": "Bearer"
}
```


2.2.4. Bước 4: Sử dụng access token để truy nhập đến resource

Để list các mail của hoangphhut@gmail.com:

```
curl -H "Authorization: Bearer
ya29.Glv8Bq4uWFgeM2h5aF1Y_LpNfc4XHpmIOTzhYm_6jmUKnuRoOiKPQambJnBKIj8wp4dmmyckFXDsSLh
64QaTQ7Kd9h1u0t6FSNkeFrBDhJPK-amK84ai3zRb639Y"
"https://www.googleapis.com/gmail/v1/users/hoangphhut@gmail.com/messages"
```

Kết quả trả về như bên dưới. Mỗi mail gồm có id và threadId:


```
{
 "messages": [
  {
   "id": "16a726c1fac73731",
   "threadId": "16a726c1fac73731"
  },
  {
   "id": "16a71d6bf9811e02",
   "threadId": "16a6f44d2453c7cc"
  },
  ...
}
```


Để xem nội dung chi tiết một mail:

```
curl -H "Authorization: Bearer
ya29.Glv8Bq4uWFgeM2h5aF1Y_LpNfc4XHpmIOTzhYm_6jmUKnuRoOiKPQambJnBKIj8wp4dmmyckFXDsSLh64Qa
TQ7Kd9h1u0t6FSNkeFrBDhJPK-amK84ai3zRb639Y"
"https://www.googleapis.com/gmail/v1/users/hoangphhut@gmail.com/messages/16933be66b9b3cd
2"
```

Kết quả trả về là thông tin của mail cùng nội dung:


```
{
 "id": "16a726c1fac73731",
 "threadId": "16a726c1fac73731",
 "labelIds": [
  "UNREAD",
  "CATEGORY_PERSONAL",
  "INBOX"
 ],
 "snippet": "Test OAuth Client was granted access to your Google Account
hoangphhut@gmail.com If you did not grant access, you should check this activity and
secure your account. Check activity You received this",
 "historyId": "1907331",
 "internalDate": "1556697848000",
"payload": {
```

```
  "partId": "",
  "mimeType": "multipart/alternative",
  "filename": "",
  "headers": [
   {
    "name": "Delivered-To",
    "value": "hoangphhut@gmail.com"
   },
   {
    "name": "Received",
    "value": "by 2002:ac8:88a:0:0:0:0:0 with SMTP id v10csp5879957qth;        Wed, 1 May
2019 01:04:09 -0700 (PDT)"
   },
   {
    "name": "X-Received",
    "value": "by 2002:ac8:3329:: with SMTP id t38mr29983895qta.350.1556697849396;
Wed, 01 May 2019 01:04:09 -0700 (PDT)"
   },
...
```


2.3. Xây dựng ứng dụng Web với JSP xác thực bằng Google Sign-in

https://developers.google.com/identity/protocols/OpenIDConnect

Thực hiện các bước như trong Youtube "Enable Google login using JAVA J2EE for websites".

Demo cho phép sử dụng Google account để login cho một ứng dụng Web (Web app). Sau khi login bằng Google acocunt, user làm việc trong session của mình (Web app hiển thị các thông tin profile của user lấy từ Google account). Web app cũng có thể lưu trữ các data riêng của nghiệp vụ Web app, liên kết với user bằng Gmail ID, như là báo cáo tài chính, lịch trình làm việc, v.v.. và khi user đã vào session của mình thì hiển thị cho user làm việc với các data này.

2.3.1. Bước 1: Đăng ký App với Google

Tiếp tục sử dụng Google project và credential đã dùng trong tutorial trước, nhưng cần bổ sung thêm một số option để hỗ trợ Google Sign-in:

- Vào https://console.developers.google.com & chọn project "Test Oauth"
- Tìm đến Credentials "Test OAuth Client" và Edit để thay đổi một số thông số:
  - Bổ sung "http://localhost:8080" trong mục Authorized JavaScript origins. Cái này cho phép Google JavaScript được chạy trong trang web "http://localhost:8080" (đây là trang Web có chứa button "Google Sign-in").
  - Bổ sung "http://localhost:8080/Google_OAuth/session.jsp" trong mục Authorized redirect URIs. Đây là nơi Web app server sẽ "nghe" Google Auth server trả kết quả về sau khi user sign-in.

2.3.2. Bước 2: Tạo ứng dụng Java Web App với trang Google Sign-in

- Tạo Java project kiểu Dynamic Web với Tomcat 7.0
- Cần download & import các Java library sau:

```
<%@ page import="org.apache.http.Header"%>
<%@ page import="org.apache.http.HttpEntity"%>
<%@ page import="org.apache.http.client.ResponseHandler"%>
<%@ page import="org.apache.http.client.methods.CloseableHttpResponse"%>
<%@ page import="org.apache.http.client.methods.HttpPost"%>
<%@ page import="org.apache.http.client.methods.RequestBuilder"%>
<%@ page import="org.apache.http.entity.ContentType"%>
<%@ page import="org.apache.http.entity.StringEntity"%>
<%@ page import="org.apache.http.impl.client.CloseableHttpClient"%>
<%@ page import="org.apache.http.impl.client.HttpClients"%>
```

```
<%@ page import="org.apache.http.message.BasicHeader"%>
<%@ page import="org.apache.http.util.EntityUtils"%>
<%@ page import="org.apache.commons.codec.binary.Base64"%>

<%@ page import="com.fasterxml.jackson.databind.JsonNode"%>
<%@ page import="com.fasterxml.jackson.databind.ObjectMapper"%>
<%@ page import="com.google.gson.JsonArray"%>
<%@ page import="com.google.gson.JsonElement"%>
<%@ page import="com.google.gson.JsonObject"%>
<%@ page import="com.google.gson.JsonParser"%>

<%@ page import="java.util.ArrayList"%>
<%@ page import="java.util.Arrays"%>
<%@ page import="java.util.List"%>
```

- Tạo file *login.jsp* chứa link "Google Sign-in" để xác thực với Google. Như trong tutorial trước, link Google Sign-in này có nhiệm vụ redirect đến trang https://accounts.google.com cùng với các tham số client ID, scope & redirect_uri để yêu cầu user login và cho phép Web app truy nhập đến các thông tin của user.

```
<html>
<head><title>Google OAuth: Sign-in</title></head>
<body>
        <a href="https://accounts.google.com/o/oauth2/v2/auth?client_id=404871172965-
f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com&amp;response_type=code&amp;scope=openid%20email&a
mp;redirect_uri=http%3A%2F%2Flocalhost%3A8080%2FGoogle_OAuth%2Fsession.jsp">Google Sign-in</a>
</body>
</html>
```

Có thể bổ sung tham số *login_hint=hoangphhut@gmail.com* vào link "Google Sign-in" để báo cho Google Auth biết muốn login với account hoangphhut@gmail.com. Nếu không có tham số này, Goole Auth sẽ luôn yêu cầu chọn account để login. Khi có tham số này và user đã login thì Google Auth bỏ qua bước chọn account để login mà trả ngay kết quả về qua redirect_uri.

2.3.3. Bước 3: Tạo trang JSP xử lý kết quả Google Auth trả về

Redirect_uri=hhttp://localhost:8080/Google_OAuth/session.jsp sẽ được Google Auth server sử dụng để trả về kết quả sau khi user sign-in. Việc cần làm là tạo trang session.JSP để xử lý kết quả này:

```
http://localhost:8080/Google_OAuth/session.jsp?code=4/PwGOqKDNDaEXYLHWfdSECCCPlvvpJr
jdZSDEIYwWT5KIotGvIb7_TjxYAl8hMpakCOuMYM_HmJ0hQc--
wQtmeq8&scope=email%20openid%20https://www.googleapis.com/auth/userinfo.email&authus
er=1&hd=soict.hust.edu.vn&session_state=0c80472655c1720c84ecf7e7a1df0321b207d817..ca
c0&prompt=consent
```

- Lấy các giá trị Google Sign-in trả về và hiển thị lên trang Web:

```
<%
String code = (String)request.getParameter("code");
String scope = (String)request.getParameter("scope");
String session_state = (String)request.getParameter("session_state");
String prompt = (String)request.getParameter("prompt");
%>
<h3>Step 1: Google Sign-in result</h3>
code: <%=code %><br>
scope: <%=scope %><br>
session_state: <%=session_state %><br>
prompt: <%=prompt %><br>
```

- Gửi code lên Google OAuth server để lấy access token:

```
<%
String client_id = "404871172965-f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com";
String client_secret = "si0dOQrUQExJZvRXrYqvd6zV";
```

```
CloseableHttpClient httpClient = HttpClients.createDefault();

HttpPost httpPost = new HttpPost("https://www.googleapis.com/oauth2/v4/token");

httpPost.setHeader("content-type", "application/x-www-form-urlencoded");
String request_body = "grant_type=authorization_code&" +
                      "code=" + code +
                      "&client_id=404871172965-f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com" +
                      "&client_secret=si0dOQrUQExJZvRXrYqvd6zV" +
                      "&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2FGoogle_OAuth%2Fsession.jsp";
StringEntity entity = new StringEntity(request_body);
httpPost.setEntity(entity);

CloseableHttpResponse resp = httpClient.execute(httpPost);
String return_body = EntityUtils.toString(resp.getEntity());
JsonParser parser = new JsonParser();
JsonElement jsonTree = parser.parse(return_body);
String access_token = jsonTree.getAsJsonObject().get("access_token").toString();
String expires_in = jsonTree.getAsJsonObject().get("expires_in").toString();
String scope2 = jsonTree.getAsJsonObject().get("scope").toString();
String token_type = jsonTree.getAsJsonObject().get("token_type").toString();
String id_token = jsonTree.getAsJsonObject().get("id_token").toString();
%>
```

- Hiển thị kết quả trả về từ Google OAuth server:

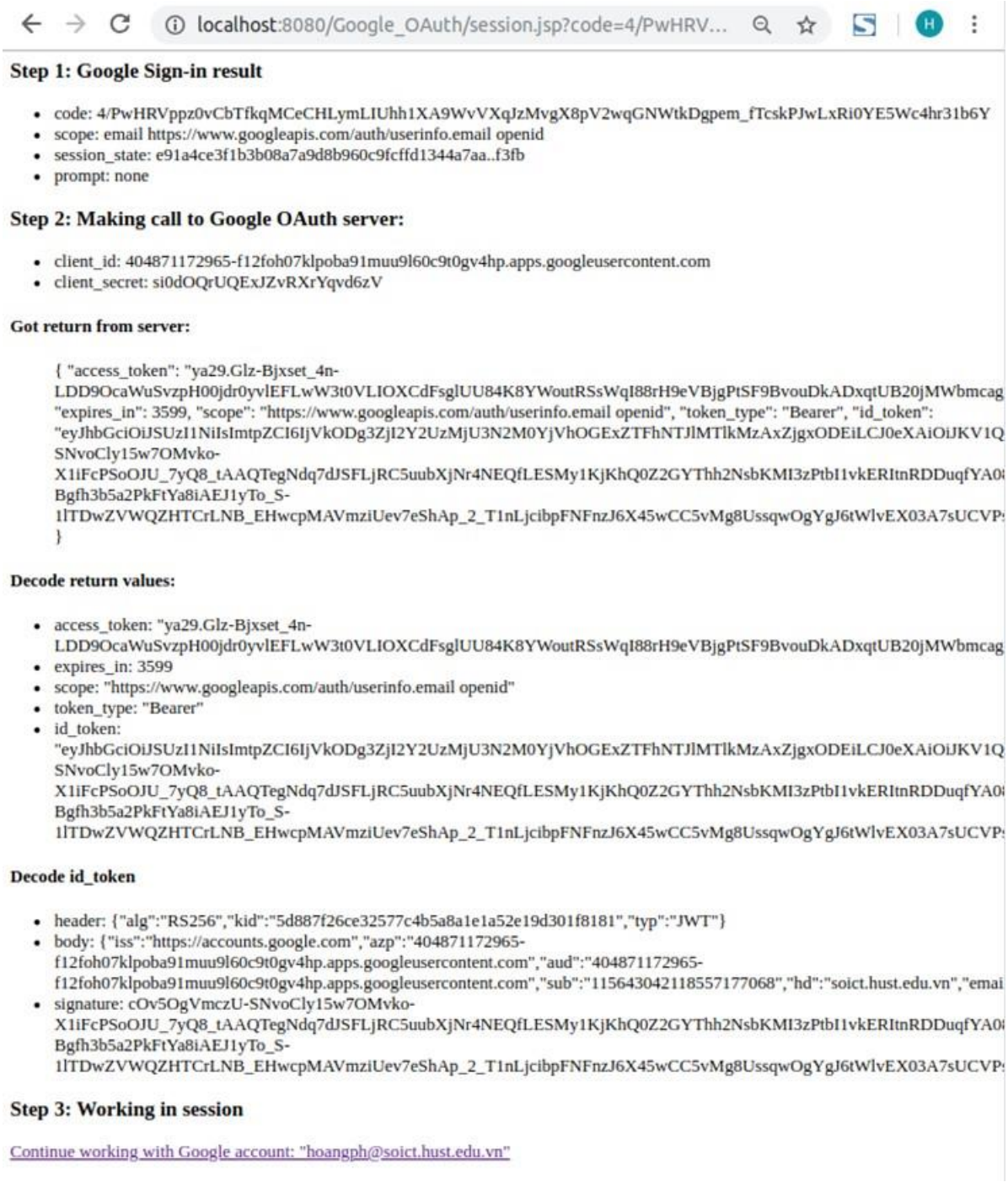
```
<h3>Step 2: Making call to Google OAuth server:</h3>
<ul>
<li>client_id: <%= client_id %></li>
<li>client_secret: <%= client_secret %></li>
</ul>

<h4>Got return from server:</h4>
<ul>
<%= return_body %>
</ul>

<h4>Decode return values:</h4>
<ul>
<li>access_token: <%= access_token %></li>
<li>expires_in: <%= expires_in %></li>
<li>scope: <%= scope2 %></li>
<li>token_type: <%= token_type %></li>
<li>id_token: <%= id_token %></li>
</ul>
<%
        Base64 base64Url = new Base64(true);
        String[] split_string = id_token.split("\\.");
        String header = new String(base64Url.decode(split_string[0]));
        String body = new String(base64Url.decode(split_string[1]));
        String signature = new String(base64Url.decode(split_string[2]));
%>
<h4>Decode id_token</h4>
<ul>
<li>header: <%= header %></li>
<li>body: <%= body %></li>
<li>signature: <%= signature %></li>
</ul>
```

- Web app server hiểu là user đã login thành công và có thể lấy một số thông tin user trực tiếp từ id_token hoặc sử dụng access_token để đọc các thông tin về Gooogle profile của user (với điều liện user cho phép). Gmail của user có thể sử dụng để làm key khi truy suất các data liên quan đến user này trong Web app hiện tại. Session ID cũng được khởi tạo và đưa vào cookie để giữ trạng thái session với Client. Trong ví dụ bên dưới sử dụng ngay email làm session_id nhưng thực tế nên tạo một chuỗi giá trị ngẫu nhiên.

```
<%
        jsonTree = parser.parse(body);
        String email = jsonTree.getAsJsonObject().get("email").toString();
%>
<a href="work.jsp">Continue working with Google account: <%= email%></a>
<%
        Cookie cookie = new Cookie("session_id",email);
```

```
            cookie.setMaxAge(60*60*24);
            response.addCookie(cookie);
%>
```

Kết quả chạy chương trình như sau:



2.3.4. Bước 4: Tạo trang JSP làm việc trong session

Trang JSP work.jsp hiển thị giá trị session_id mà user đang làm việc, đồng thời cung cấp link "Logout" để user thoát khỏi phiên làm việc (bằng cách xóa cookie session_id đi). Nếu user vào trực tiếp trang web này để làm việc, nó sẽ kiểm tra session_id còn giá trị không. Trường hợp session_id đã hết hiệu lực, trang web này hiển thị thông báo và yêu cầu user quay về trang login:

```
<<html><head><title>Google OAuth: Working in session</title></head>
<body>
```

```
<%
        Cookie[] cookies = request.getCookies();
        String logout = (String)request.getParameter("logout");
        if (logout != null) {
                for (int i=0; i<cookies.length; i++) {
                        if (cookies[i].getName().compareTo("session_id") == 0) {
                                cookies[i].setMaxAge(0);
                                response.addCookie(cookies[i]);
                                response.sendRedirect("login.jsp");
                                return;
                        }
                }
        }

        boolean session_validate = false;
        for (int i=0; i<cookies.length; i++) {
                if (cookies[i].getName().compareTo("session_id") == 0) {
                        session_validate = true;
%>
<h3>You are working in a session identified by following cookie:</h3>
<%= cookies[i].getName() %>: <%= cookies[i].getValue() %>
<p>
<a href="work.jsp?logout=yes">Logout</a>
<%
                }
        }

        if(!session_validate) {
%>
<h3>Your session has terminated</h3>
Please <a href="login.jsp">login</a> again.
<%
        }
%>

</body>
</html>
```

Kết quả chạy chương trình:

Trường hợp đã logout mà vào trực tiếp trang work.jsp thì sẽ hiển thị thông báo phiên làm việc đã kết thúc:

2.3.5. Sử dụng Google library (JavaScript & Java)

Ví dụ trên xây dựng ứng dụng theo từng bước tường minh của mô hình Google Sign-in. Google ẩn đi các bước phức tạp nằng bằng cách cung cấp thư viện JavaScript để nhúng button

"Google Sign-in" và trang web và thư viên Java (và các ngôn ngữ khác) để truy nhập đến các Google API. Sau đây là ví dụ sử dụng thư viên Google JavaScript để nhúng button "Google Sign-in":

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Insert title here</title>
    <script src="https://apis.google.com/js/platform.js" async defer></script>
    <meta name="google-signin-client_id" content="COPY YOUR CLIENT ID HERE">
</head>
<body>
    <div class="g-signin2" data-onsuccess="onSignIn" id="myP"></div>
        <img id="myImg"><br>
        <p id="name"></p>
        <div id="status">
    </div>
    <script type="text/javascript">
        function onSignIn(googleUser) {
        // window.location.href='success.jsp';
        var profile = googleUser.getBasicProfile();
        var imagurl=profile.getImageUrl();
        var name=profile.getName();
        var email=profile.getEmail();
        document.getElementById("myImg").src = imagurl;
        document.getElementById("name").innerHTML = name;
        document.getElementById("myP").style.visibility = "hidden";
        document.getElementById("status").innerHTML = 'Welcome '+name+'!<a href=success.jsp?
        email='+email+'&name='+name+'/>Continue with Google login</a></p>'
    }
    </script>
    <button onclick="myFunction()">Sign Out</button>
    <script>
        function myFunction() {
        gapi.auth2.getAuthInstance().disconnect();
        location.reload();
    }
    </script>
</body>
</html>
```

3. Phụ lục

3.1. Youtube

3.1.1. How Google is using OAuth

https://www.youtube.com/watch?v=fxRXLbgX53A

Slide trình bày các khái niệm cơ bản đầu tiên của mô hình OAuth. Rất trực quan dễ hiểu bằng demo App truy nhập GMail. Bài toán đặt ra là xây dựng app để truy nhập đến gmail của người sử dụng. App cần được user (người sở hữu hòm thư) cho phép truy nhập đến Gmail của mình.

Video có 3 phần:
- lý thuyết
- chuẩn bị môi trường
- demo sử dụng curl

3.1.2. OAuth 2.0 and OpenID Connect (in plain English)

https://www.youtube.com/watch?v=0VWkQMr7r_c

Trình bày nhiều vễ kỹ thuật và lý thuyết hơn clip bên trên (overview và demo đơn giản). Theo dõi kỹ có thể hiểu khá đầy đủ về các ngóc ngách của mô hình Oauth.

3.1.3. Oktane18: Using OAuth and OpenID Connect in Your Applications

https://www.youtube.com/watch?v=rZRwE3d-gm0

Chưa xem cái này nhưng có vẻ trình bày cũng khá chi tiết về mặt kỹ thuật như Youtube bên trên.

3.1.4. Enable Google login using JAVA J2EE for websites

https://www.youtube.com/watch?v=fhE2kEiop6c

Hướng dẫn các bước tạo Web app với JSP cho phép login bằng Google Account và truy nhập đến các thông tin user sau khi đã login. Demo cũng mô tả button <logout> để kết thúc phiên làm việc với một Google Account.

3.2. Specification (chuẩn kỹ thuật)

- RFC 6749: The OAuth 2.0 Authorization Framework: https://tools.ietf.org/html/rfc6749
- OpenID: https://openid.net

3.3. Khác

- https://developers.google.com/identity/sign-in/web/sign-in
- https://256stuff.com/gray/docs/oauth2.0/
- https://jwt.io/ (tool decode JWT - JSON Web Token, được sử dụng cho id_token khi làm Google Sign-in)

3.4. Mã nguồn

3.4.1. login.jsp

```
<html>
<head><title>Google OAuth: Sign-in</title></head>
<body>
        <a href="https://accounts.google.com/o/oauth2/v2/auth?client_id=404871172965-
f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com&amp;response_type=code&amp;scope=openid%20email&a
mp;redirect_uri=http%3A%2F%2Flocalhost%3A8080%2FGoogle_OAuth%2Fsession.jsp">Google Sign-in</a>
</body>
</html>
```

3.4.2. session.jsp

```
<%@ page import="org.apache.http.Header"%>
<%@ page import="org.apache.http.HttpEntity"%>
<%@ page import="org.apache.http.client.ResponseHandler"%>
<%@ page import="org.apache.http.client.methods.CloseableHttpResponse"%>
<%@ page import="org.apache.http.client.methods.HttpPost"%>
<%@ page import="org.apache.http.client.methods.RequestBuilder"%>
<%@ page import="org.apache.http.entity.ContentType"%>
<%@ page import="org.apache.http.entity.StringEntity"%>
<%@ page import="org.apache.http.impl.client.CloseableHttpClient"%>
<%@ page import="org.apache.http.impl.client.HttpClients"%>
<%@ page import="org.apache.http.message.BasicHeader"%>
<%@ page import="org.apache.http.util.EntityUtils"%>
<%@ page import="org.apache.commons.codec.binary.Base64"%>
```

```
<%@ page import="java.util.ArrayList"%>
<%@ page import="java.util.Arrays"%>
<%@ page import="java.util.List"%>

<%@ page import="com.fasterxml.jackson.databind.JsonNode"%>
<%@ page import="com.fasterxml.jackson.databind.ObjectMapper"%>
<%@ page import="com.google.gson.JsonArray"%>
<%@ page import="com.google.gson.JsonElement"%>
<%@ page import="com.google.gson.JsonObject"%>
<%@ page import="com.google.gson.JsonParser"%>

<html>
<head>
<title>Google OAuth: Session initialization after Google Sign-in</title>
</head>
<body>

<%
        String code = (String)request.getParameter("code");
        String scope = (String)request.getParameter("scope");
        String session_state = (String)request.getParameter("session_state");
        String prompt = (String)request.getParameter("prompt");
%>
<h3>Step 1: Google Sign-in result</h3>
<ul>
<li>code: <%=code %></li>
<li>scope: <%=scope %></li>
<li>session_state: <%=session_state %></li>
<li>prompt: <%=prompt %></li>
</ul>
<%
        String client_id = "404871172965-f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com";
        String client_secret = "si0dOQrUQExJZvRXrYqvd6zV";

        CloseableHttpClient httpClient = HttpClients.createDefault();

        HttpPost httpPost = new HttpPost("https://www.googleapis.com/oauth2/v4/token");

        httpPost.setHeader("content-type", "application/x-www-form-urlencoded");
        String request_body = "grant_type=authorization_code&" +
                "code=" + code +
                "&client_id=404871172965-f12foh07klpoba91muu9l60c9t0gv4hp.apps.googleusercontent.com" +
                "&client_secret=si0dOQrUQExJZvRXrYqvd6zV" +
                "&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2FGoogle_OAuth%2Fsession.jsp";
        StringEntity entity = new StringEntity(request_body);
    httpPost.setEntity(entity);

  //String responseBody = httpClient.execute(httpPost, responseHandler);
    CloseableHttpResponse resp = httpClient.execute(httpPost);
    String return_body = EntityUtils.toString(resp.getEntity());
  JsonParser parser = new JsonParser();
    JsonElement jsonTree = parser.parse(return_body);
  String access_token = jsonTree.getAsJsonObject().get("access_token").toString();
  String expires_in = jsonTree.getAsJsonObject().get("expires_in").toString();
  String scope2 = jsonTree.getAsJsonObject().get("scope").toString();
  String token_type = jsonTree.getAsJsonObject().get("token_type").toString();
  String id_token = jsonTree.getAsJsonObject().get("id_token").toString();
%>

<h3>Step 2: Making call to Google OAuth server:</h3>
<ul>
<li>client_id: <%= client_id %></li>
<li>client_secret: <%= client_secret %></li>
</ul>

<h4>Got return from server:</h4>
<ul>
<%= return_body %>
</ul>

<h4>Decode return values:</h4>
<ul>
<li>access_token: <%= access_token %></li>
<li>expires_in: <%= expires_in %></li>
<li>scope: <%= scope2 %></li>
<li>token_type: <%= token_type %></li>
<li>id_token: <%= id_token %></li>
</ul>
<%
```

```
            Base64 base64Url = new Base64(true);
            String[] split_string = id_token.split("\\.");
            String header = new String(base64Url.decode(split_string[0]));
            String body = new String(base64Url.decode(split_string[1]));
            String signature = split_string[2];
%>
<h4>Decode id_token</h4>
<ul>
<li>header: <%= header %></li>
<li>body: <%= body %></li>
<li>signature: <%= signature %></li>
</ul>

<h3>Step 3: Working in session</h3>
<%
            jsonTree = parser.parse(body);
            String email = jsonTree.getAsJsonObject().get("email").toString();
%>
<a href="work.jsp">Continue working with Google account: <%= email%></a>
<%
            Cookie cookie = new Cookie("session_id",email);
            cookie.setMaxAge(60*60*24);
            response.addCookie(cookie);
%>
</body>
</html>
```

3.4.3. work.jsp

```
<html><head><title>Google OAuth: Working in session</title></head>
<body>

<%
            Cookie[] cookies = request.getCookies();
            String logout = (String)request.getParameter("logout");
            if (logout != null) {
                        for (int i=0; i<cookies.length; i++) {
                                    if (cookies[i].getName().compareTo("session_id") == 0) {
                                                cookies[i].setMaxAge(0);
                                                response.addCookie(cookies[i]);
                                                response.sendRedirect("login.jsp");
                                                return;
                                    }
                        }
            }

            boolean session_validate = false;
            for (int i=0; i<cookies.length; i++) {
                        if (cookies[i].getName().compareTo("session_id") == 0) {
                                    session_validate = true;
%>
<h3>You are working in a session identified by following cookie:</h3>
<%= cookies[i].getName() %>: <%= cookies[i].getValue() %>
<p>
<a href="work.jsp?logout=yes">Logout</a>
<%
                        }
            }

            if (!session_validate) {
%>
<h3>Your session has terminated</h3>
Please <a href="login.jsp">login</a> again.
<%
            }
%>


</body>
</html>
```