

## OAuth 2.0 framework.

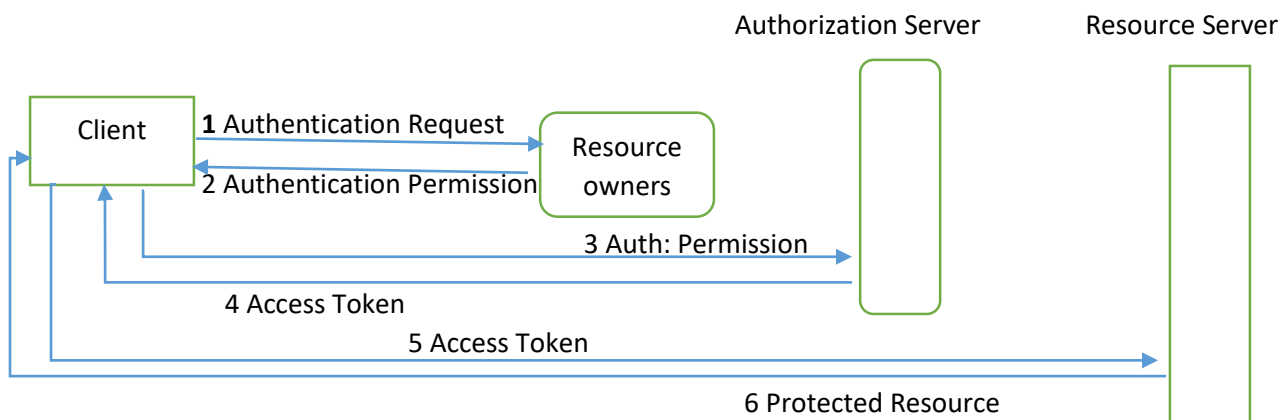
### Introduction.

OAuth 2 is an industry standard application use in web, desktop and mobile base authentication system. This will use to get the user account access on HTTP services, For example GitHub, Google drive, Face Book. This will enable the user access to defined servers and application through the user accounts [1].

In mobile apps OAuth is use for Native and Browser Base Apps and device authentication process. From the token management process will use for determine the activity state and meta information of token revocation process and JOSE web token process [2].

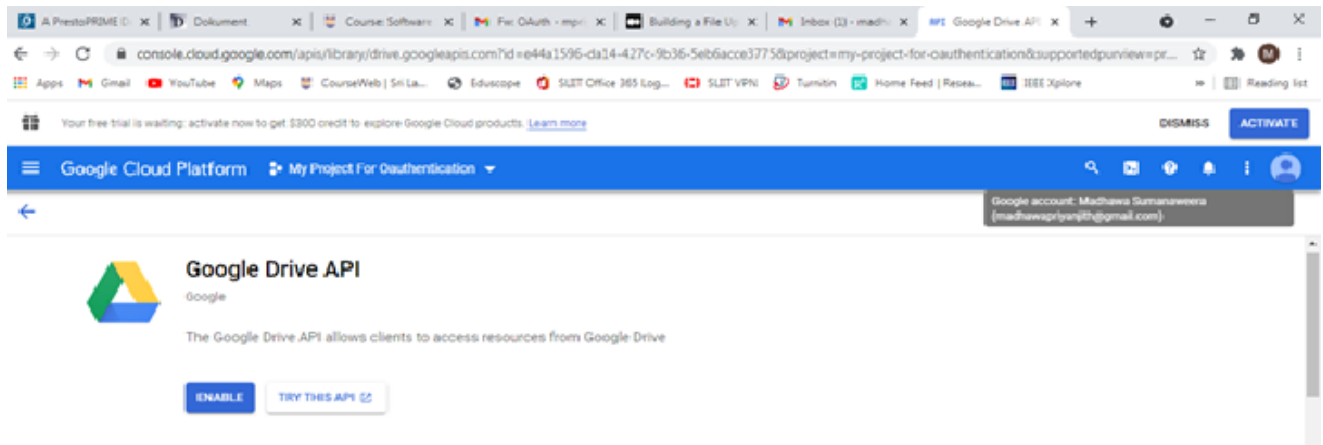
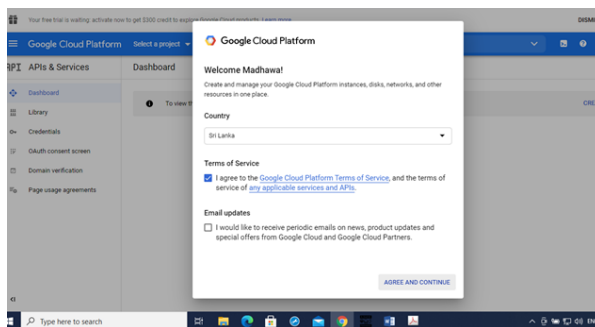
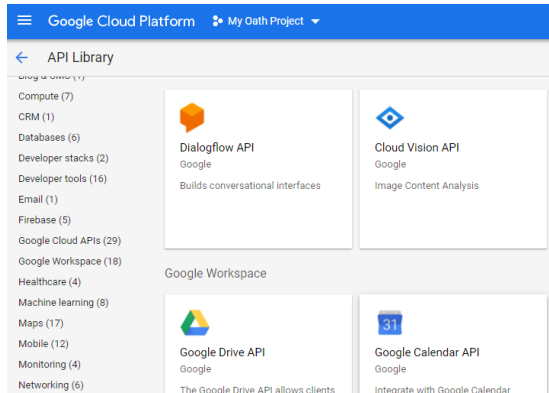
In this assignment I used Java for develop the OAuth application and will explain, how the basic background arrangement in Google Drive API to get the access to Google Drive by authenticating through Java application and use of the OAuth 2 services use for get access to Google drive through. This application is using Spring Boot and the Google Client Library for OAuth and Drive

Below diagram explain the high level process OAuth application [3].



## Key Arrangements.

In the initial process, need to setup a project in Google API console to get the obtain OAuth 2.0 credentials and below are the steps of creating this in Google API.



Browser tabs: A PrestoPRIME D..., Dokument, Course: Software..., Fw: OAuth - mpr..., Building a File Up..., Inbox (1) - madh..., New Project - G...

console.cloud.google.com/projectcreate?previousPage=%2Fprojectselector%2Fapis%2Fdashboard%3Fpl%3D1%26supportedpurview%3Dproject&supportedpurview=proj...

Apps Gmail YouTube Maps CourseWeb | Sri La... Eduscope SLIT Office 365 Log... SLIT VPN Turnitin Home Feed | Resea... IEEE Xplore Reading list

Your free trial is waiting: activate now to get \$300 credit to explore Google Cloud products. [Learn more](#) **DISMISS** **ACTIVATE**

**Google Cloud Platform** Search products and resources

**New Project** Google account: Madhawa Sumanaweera (madhawapriyanjith@gmail.com)

You have 12 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)  
[MANAGE QUOTAS](#)

Project name \*  
My Project For OAuthentication ?

Project ID: my-project-for-oauthentication. It cannot be changed later. [EDIT](#)

Location \*  
No organization [BROWSE](#)

Parent organization or folder

**CREATE** **CANCEL**

Below is the credential Obtained for my OAuth project and the client ID and other details related to this OAuth project.

← → ↻ <https://console.cloud.google.com/apis/credentials/oauthclient/20987090643-1ni3blbue0k!>

**Google Cloud Platform** My Oath Project Search products and resources

**API** APIs & Services Client ID for Web application [DOWNLOAD JSON](#) [RESE](#)

Dashboard  
 Library  
 **Credentials**  
 OAuth consent screen  
 Domain verification  
 Page usage agreements

**Name \***  
Web client 1

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#).

**Authorized JavaScript origins ?**

For use with requests from a browser

Google Cloud Platform

My Oath Project

Search products and resources

APIs & Services

Dashboard

Library

Credentials

OAuth consent screen

Domain verification

Page usage agreements

Credentials

+ CREATE CREDENTIALS

DELETE

Create credentials to access your enabled APIs. [Learn more](#)

API Keys

<input type="checkbox"/>	Name	Creation date ↓	Restrictions	Key
No API keys to display				

OAuth 2.0 Client IDs

<input type="checkbox"/>	Name	Creation date ↓	Type	Client ID	
<input type="checkbox"/>	Web client 1	Apr 6, 2021	Web application	20987090643-1ni3b...	<div></div> <div></div> <div></div>

Client ID	20987090643-1ni3blbue0k5m6922ei04r2tqutlh4on.apps.googleusercontent.com
Client secret	wVucRVI-tB9QGuaNjGmHUAkj
Creation date	April 6, 2021 at 5:19:10 PM GMT+5

## OAuth Consent Screen.

In this page the details of the author of the application will added for end user references.

API	APIs & Services	Edit app registration
	<ul style="list-style-type: none"><li>Dashboard</li><li>Library</li><li>Credentials</li><li><b>OAuth consent screen</b></li><li>Domain verification</li><li>Page usage agreements</li></ul>	<div><b>1 OAuth consent screen</b> — <b>2</b> Scopes — <b>3</b> Test users — <b>4</b> Sum</div> <h3>App information</h3> <p>This shows in the consent screen, and helps end users know who you are and contact you</p> <div><b>App name *</b> My Oath APP <small>The name of the app asking for consent</small></div> <div><b>User support email *</b> madhawapriyanjith@gmail.com <small>For users to contact you with questions about their consent</small></div>

The screenshot shows the Google Cloud Platform console interface. The left sidebar contains a menu with 'APIs & Services' selected, and sub-items: 'Dashboard', 'Library', 'Credentials', 'OAuth consent screen', 'Domain verification', and 'Page usage agreements'. The main content area is titled 'Add credentials to your project' and contains a wizard with the following steps:

- Find out what kind of credentials you need**
  - Which API are you using?  
Google Drive API
  - Where will you be calling the API from?  
Web browser (JavaScript)
  - What data will you be accessing?  
☒ User data  
☐ Application data

## JASON Token.

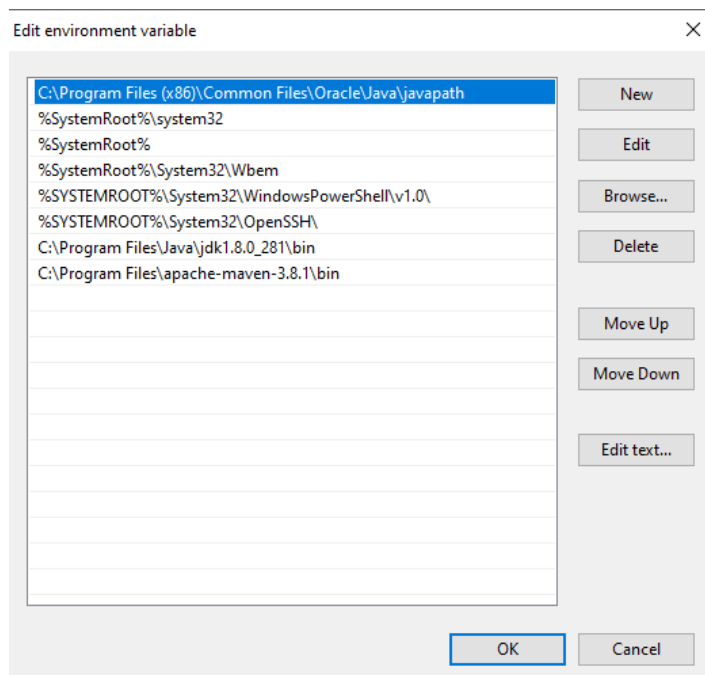
Below is the Jason token generated for this application.

```
{"web":{"client_id":"20987090643-1ni3blbue0k5m6922ei04r2tqutlh4on.apps.googleusercontent.com","project_id":"my-oath-project-308700","auth_uri":"https://accounts.google.com/o/oauth2/auth","token_uri":"https://oauth2.googleapis.com/token","auth_provider_x509_cert_url":"https://www.googleapis.com/oauth2/v1/certs","client_secret":"wVucRVI-tB9QGuaNjGmHUAkj"}}
```

Once the details are filled save before exit.

## Application Processing

For develop this application used JDK 1.8 and Maven 3.8.1 Software. As an pre arrangement modify the environment variable as below for JDK and maven application path.



After the coding is completed (Please refer the Appendix) we are able to process the coding and Google authentication process.

This PC > New Volume (D:) > oauth-app

Name	Date modified	Type	Size
.mvn	4/15/2021 2:51 PM	File folder	
.settings	4/6/2021 7:24 PM	File folder	
bin	4/6/2021 4:55 PM	File folder	
src	4/15/2021 11:09 AM	File folder	
target	4/15/2021 2:56 PM	File folder	
.classpath	4/6/2021 7:24 PM	CLASSPATH File	2 KB
.gitignore	4/6/2021 8:57 PM	GITIGNORE File	1 KB
.project	4/6/2021 7:24 PM	PROJECT File	1 KB
pom	4/6/2021 4:55 PM	XML Document	3 KB

```

Administrator: Command Prompt - mvn clean package spring-boot:run
.\oauth-app>mvn clean package spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.munsif.ssd.oauth >-----
[INFO] Building oauth 0.0.1-SNAPSHOT
[INFO]
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:3.0.0:clean (default-clean) @ oauth ---
[INFO] Deleting D:\oauth-app\target
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ oauth ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 16 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ oauth ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 9 source files to D:\oauth-app\target\classes
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ oauth ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory D:\oauth-app\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.7.0:testCompile (default-testCompile) @ oauth ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.21.0:test (default-test) @ oauth ---
  
```

```

Spring Boot :: (v2.0.5.RELEASE)
2021-04-11 20:51:41.783 INFO 21512 --- [ restartedMain] com.madhawa.ssd.oauth.OauthApplication : Starting OauthApplication on LAPITRD16 with PID 21512 (D:\oauth-app\target\classes started by admin in D:\oauth-app)
2021-04-11 20:51:41.787 INFO 21512 --- [ restartedMain] com.madhawa.ssd.oauth.OauthApplication : No active profile set, falling back to default profiles: default
2021-04-11 20:51:41.939 INFO 21512 --- [ restartedMain] ConfigServletWebServerApplicationContext : Refreshing org.springframework.boot.web.servlet.context.AnnotationConfigServletWebServerApplicationContext@b83dbee: startup date [Sun Apr 11 20:51:41 IST 2021]; root of context hierarchy
2021-04-11 20:51:45.384 INFO 21512 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-04-11 20:51:45.439 INFO 21512 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-04-11 20:51:45.440 INFO 21512 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.34
2021-04-11 20:51:45.469 INFO 21512 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener : The APR based Apache Tomcat Native library which allows optimal performance in production environments was not found on the java.library.path: [C:\Program Files\Java\jdk1.8.0_251\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\IBM\IBM Client Access\Shared;C:\Program Files (x86)\IBM\IBM Client Access\;C:\Program Files\Git\cmd;C:\Program Files\Docker\Docker\resources\bin;C:\ProgramData\DockerDesktop\version-bin;C:\Program Files\apache-maven-3.8.1\bin;D:\python;C:\Users\Administrator\AppData\Local\Microsoft\WindowsApps;%JAVA_PATH%;C:\Users\Administrator\AppData\Local\Programs\Microsoft VS Code\bin;C:\Program Files\JetBrains\PyCharm 2020.3.2\bin;C:\Program Files\JetBrains\PyCharm Community Edition 2020.3.2\bin;...]
  
```

Prompt for select the google account for authentication

# Welcome to G Suite File Uploader

Please authorize your Google Drive by click on the button below to access your drive



ගිණුමක් තෝරාගන්න



**Madhawa Sumanaweera**  
madhawapriyanjith@gmail.com

වරන ලද

 වෙනත් ගිණුමක් භාවිත කරන්න

 ගිණුමක් ඉවත් කරන්න

 Google සමගින් පුරන්න

**Madhawa Sumanaweera**

 madhawapriyanjith@gmail.com

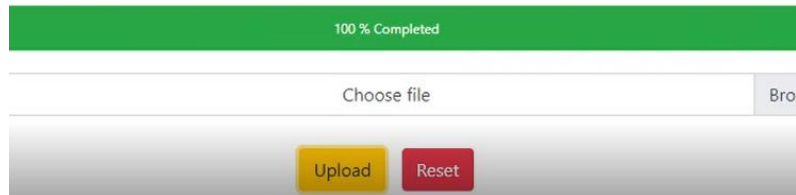
ඔබේ මුරපදය ඇතුළු කරන්න

....|

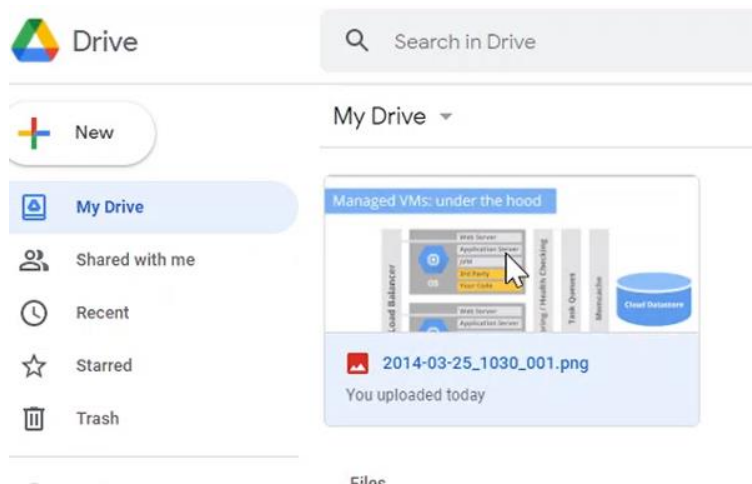


# GDrive File Upload Manager

You are now logged in via Google OAuth



## Loading of the Google Drive



## Appendix

### Code Demonstration.

Below are the Java codes used for this OAuth Development.

#### # Application Content

```
package com.madhawa.ssd.oauth.constant;

import java.util.Collections;
import java.util.List;

import com.google.api.client.http.HttpTransport;
import com.google.api.client.http.javanet.NetHttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.services.drive.DriveScopes;

public class ApplicationConstant {

    public static HttpTransport HTTP_TRANSPORT = new NetHttpTransport();
    public static JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();

    public static final List<String> SCOPES = Collections.singletonList(DriveScopes.DRIVE);

    public static final String USER_IDENTIFIER_KEY = "MY_TEST_USER";
    public static final String APPLICATION_NAME = "SSD OAuth Spring App";
    public static final String PARENT_FOLDER_NAME = "OAuth Demo App Uploaded";
}
```

## **# Main Controller.**

```
package com.madhawa.ssd.oauth.controller;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.multipart.MultipartFile;

import com.madhawa.ssd.oauth.model.UploadFile;
import com.madhawa.ssd.oauth.service.AuthorizationService;
import com.madhawa.ssd.oauth.service.DriveService;

@Controller
public class MainController {

    private Logger logger = LoggerFactory.getLogger(MainController.class);

    @Autowired
    AuthorizationService authorizationService;

    @Autowired
    DriveService driveService;
```

```

/**
 * Handles the root request. Checks if user is already authenticated via SSO.
 *
 * @return
 * @throws Exception
 */
@GetMapping("/")
public String showHomePage() throws Exception {
    if (authorizationService.isUserAuthenticated()) {
        logger.debug("User is authenticated. Redirecting to home...");
        return "redirect:/home";
    } else {
        logger.debug("User is not authenticated. Redirecting to sso...");
        return "redirect:/login";
    }
}

```

```

/**
 * Directs to login
 *
 * @return
 */
@GetMapping("/login")
public String goToLogin() {
    return "index.html";
}

```

```

/**

```

```

* Directs to home
*
* @return
*/
@GetMapping("/home")
public String goToHome() {
    return "home.html";
}

/**
* Calls the Google OAuth service to authorize the app
*
* @param response
* @throws Exception
*/
@GetMapping("/googlesignin")
public void doGoogleSignIn(HttpServletResponse response) throws Exception {
    logger.debug("SSO Called...");
    response.sendRedirect(authorizationService.authenticateUserViaGoogle());
}

/**
* Applications Callback URI for redirection from Google auth server after user
* approval/consent
*
* @param request
* @return
* @throws Exception
*/

```

```

@GetMapping("/oauth/callback")

public String saveAuthorizationCode(HttpServletRequest request) throws Exception {

    logger.debug("SSO Callback invoked...");

    String code = request.getParameter("code");

    logger.debug("SSO Callback Code Value..., " + code);

    if (code != null) {

        authorizationService.exchangeCodeForTokens(code);

        return "home.html";

    }

    return "index.html";

}

```

```

/**
 * Handles logout
 *
 * @return
 * @throws Exception
 */

@GetMapping("/logout")

public String logout(HttpServletRequest request) throws Exception {

    logger.debug("Logout invoked...");

    authorizationService.removeUserSession(request);

    return "redirect:/login";

}

```

```

/**
 * Handles the files being uploaded to GDrive
 *

```

```

    * @param request
    * @param uploadedFile
    * @return
    * @throws Exception
    */
    @PostMapping("/upload")
    public String uploadFile(HttpServletRequest request, @ModelAttribute UploadFile uploadedFile)
    throws Exception {
        MultipartFile multipartFile = uploadedFile.getMultipartFile();
        driveService.uploadFile(multipartFile);
        return "redirect:/home?status=success";
    }
}

```

## # Upload File

```

package com.madhawa.ssd.oauth.model;

import org.springframework.web.multipart.MultipartFile;

public class UploadFile {

    // private static final long serialVersionUID = 1L;
    private MultipartFile multipartFile;

    public MultipartFile getMultipartFile() {
        return multipartFile;
    }
}

```

```
        public void setMultipartFile(MultipartFile multipartFile) {  
            this.multipartFile = multipartFile;  
        }  
    }  
}
```

## **# Authorization Service**

```
package com.madhawa.ssd.oauth.service.impl;
```

```
import java.io.IOException;
```

```
import java.io.InputStreamReader;
```

```
import javax.annotation.PostConstruct;
```

```
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpSession;
```

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Service;
```

```
import com.google.api.client.auth.oauth2.Credential;
```

```
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
```

```
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeRequestUrl;
```

```
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
```

```
import com.google.api.client.googleapis.auth.oauth2.GoogleTokenResponse;
```

```
import com.google.api.client.util.store.FileDataStoreFactory;
```

```
import com.madhawa.ssd.oauth.constant.ApplicationConstant;
```

```
import com.madhawa.ssd.oauth.service.AuthorizationService;
```

```
import com.madhawa.ssd.oauth.util.ApplicationConfig;
```



@Service

```
public class AuthorizationServiceImpl implements AuthorizationService {
```

```
    private Logger logger = LoggerFactory.getLogger(AuthorizationServiceImpl.class);
```

```
    private GoogleAuthorizationCodeFlow flow;
```

```
    private FileDataStoreFactory dataStoreFactory;
```

@Autowired

```
    private ApplicationConfig config;
```

@PostConstruct

```
    public void init() throws Exception {
```

```
        InputStreamReader reader = new  
        InputStreamReader(config.getDriveSecretKeys().getInputStream());
```

```
        dataStoreFactory = new FileDataStoreFactory(config.getCredentialsFolder().getFile());
```

```
        GoogleClientSecrets clientSecrets =  
        GoogleClientSecrets.load(ApplicationConstant.JSON_FACTORY, reader);
```

```
        flow = new  
        GoogleAuthorizationCodeFlow.Builder(ApplicationConstant.HTTP_TRANSPORT,  
        ApplicationConstant.JSON_FACTORY, clientSecrets,
```

```
        ApplicationConstant.SCOPEES).setDataStoreFactory(dataStoreFactory).build();
```

```
    }
```

@Override

```
    public boolean isUserAuthenticated() throws Exception {
```

```
        Credential credential = getCredentials();
```

```
        if (credential != null) {
```

```

        boolean isValidToken = credential.refreshToken();

        logger.debug("isValidToken, " + isValidToken);

        return isValidToken;
    }

    return false;
}

```

@Override

```

public Credential getCredentials() throws IOException {

    return flow.loadCredential(ApplicationConstant.USER_IDENTIFIER_KEY);

}

```

@Override

```

public String authenticateUserViaGoogle() throws Exception {

    GoogleAuthorizationCodeRequestUrl url = flow.newAuthorizationUrl();

    String redirectUrl =
url.setRedirectUri(config.getCALLBACK_URI()).setAccessType("offline").build();

    logger.debug("redirectUrl, " + redirectUrl);

    return redirectUrl;

}

```

@Override

```

public void exchangeCodeForTokens(String code) throws Exception {

    // exchange the code against the access token and refresh token

    GoogleTokenResponse tokenResponse =
flow.newTokenRequest(code).setRedirectUri(config.getCALLBACK_URI()).execute();

    flow.createAndStoreCredential(tokenResponse,
ApplicationConstant.USER_IDENTIFIER_KEY);

}

```

@Override

```

        public void removeUserSession(HttpServletRequest request) throws Exception {
//            HttpSession session = request.getSession(false);
//            session = request.getSession(true);
//            if (session != null) {
//                session.invalidate();
//                logger.info("Logged Out...");
//            }

            // revoke token and clear the local storage
            dataStoreFactory.getDataStore(config.getCredentialsFolder().getFilename()).clear();
        }
    }
}

```

#### **# Drive Service**

```

package com.madhawa.ssd.oauth.service.impl;

import javax.annotation.PostConstruct;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.http.FileContent;
import com.google.api.services.drive.Drive;
import com.google.api.services.drive.model.File;
import com.madhawa.ssd.oauth.constant.ApplicationConstant;

```

```

import com.madhawa.ssd.oauth.service.AuthorizationService;
import com.madhawa.ssd.oauth.service.DriveService;
import com.madhawa.ssd.oauth.util.ApplicationConfig;

@Service
public class DriveServiceImpl implements DriveService {

    private Logger logger = LoggerFactory.getLogger(DriveServiceImpl.class);

    private Drive driveService;

    @Autowired
    AuthorizationService authorizationService;

    @Autowired
    ApplicationConfig applicationConfig;

    @PostConstruct
    public void init() throws Exception {
        Credential credential = authorizationService.getCredentials();
        driveService = new Drive.Builder(ApplicationConstant.HTTP_TRANSPORT,
ApplicationConstant.JSON_FACTORY, credential)
                .setApplicationName(ApplicationConstant.APPLICATION_NAME).build();
    }

    @Override
    public void uploadFile(MultipartFile multipartFile) throws Exception {
        logger.debug("Inside Upload Service...");
    }
}

```

```

        String path = applicationConfig.getTemporaryFolder();
        String fileName = multipartFile.getOriginalFilename();
        String contentType = multipartFile.getContentType();

        java.io.File transferredFile = new java.io.File(path, fileName);
        multipartFile.transferTo(transferredFile);

        File fileMetadata = new File();
        fileMetadata.setName(fileName);

        FileContent mediaContent = new FileContent(contentType, transferredFile);

        File file = driveService.files().create(fileMetadata,
mediaContent).setFields("id").execute();

        logger.debug("File ID: " + file.getName() + ", " + file.getId());
    }
}

```

## **# Application Config**

```

package com.madhawa.ssd.oauth.util;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.PropertySource;
import org.springframework.core.io.Resource;
import org.springframework.stereotype.Component;

```

```

@Component
@PropertySource("classpath:application.properties")
@ConfigurationProperties
public class ApplicationConfig {

    @Value("${google.oauth.callback.uri}")
    private String CALLBACK_URI;

    @Value("${google.secret.key.path}")
    private Resource driveSecretKeys;

    @Value("${google.credentials.folder.path}")
    private Resource credentialsFolder;

    @Value("${myapp.temp.path}")
    private String temporaryFolder;

    public String getCALLBACK_URI() {
        return CALLBACK_URI;
    }

    public void setCALLBACK_URI(String CALLBACK_URI_p) {
        CALLBACK_URI = CALLBACK_URI_p;
    }

    public Resource getDriveSecretKeys() {
        return driveSecretKeys;
    }
}

```

```

    public void setDriveSecretKeys(Resource driveSecretKeys) {
        this.driveSecretKeys = driveSecretKeys;
    }

    public Resource getCredentialsFolder() {
        return credentialsFolder;
    }

    public void setCredentialsFolder(Resource credentialsFolder) {
        this.credentialsFolder = credentialsFolder;
    }

    public String getTemporaryFolder() {
        return temporaryFolder;
    }

    public void setTemporaryFolder(String temporaryFolder) {
        this.temporaryFolder = temporaryFolder;
    }
}

```

### **# OAuth Application.**

```

package com.madhawa.ssd.oauth;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

```

```
public class OAuthApplication {
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(OAuthApplication.class, args);
```

```
    }
```

```
}
```

### **# Google OAuth**

```
google.secret.key.path=classpath:keys/gsuite_drive_keys.json
```

```
google.oauth.callback.uri=http://localhost:8080/home
```

```
google.credentials.folder.path=file:D:\\oauth-app\\src\\main\\resources\\credentials
```

### **# Logger Level**

```
logging.level.com.munsif.ssd.oauth=DEBUG
```

### **# Spring MultiPart File**

```
spring.servlet.multipart.max-file-size=50MB
```

```
spring.servlet.multipart.max-request-size=50MB
```

```
spring.http.multipart.enabled=true
```

```
#spring.resources.cache.cachecontrol.max-age=1
```

```
myapp.temp.path=D:\\oauth-app\\src\\main\\resources\\tempfolder
```



## Referance

- 1) <https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2>
- 2) <https://oauth.net/2/>
- 3) <https://medium.com/@munsifmusthafa03/building-a-file-upload-service-to-your-google-drive-using-oauth-2-0-d883d6d67fe8>