Use Google OpenIDConnect OAuth2

OpenIDConnect
Google OAuth2

Abstract

Google offers a Spring-centric tool to provide OAuth2 protection, called OpenIDConnect. There are three form-factors that Google's OpenIDConnect OAuth2 product supports:

Android devices
iOS devices
ßrowsers (User-Agent)

This document explores how to register the latter User Agent – a browser.

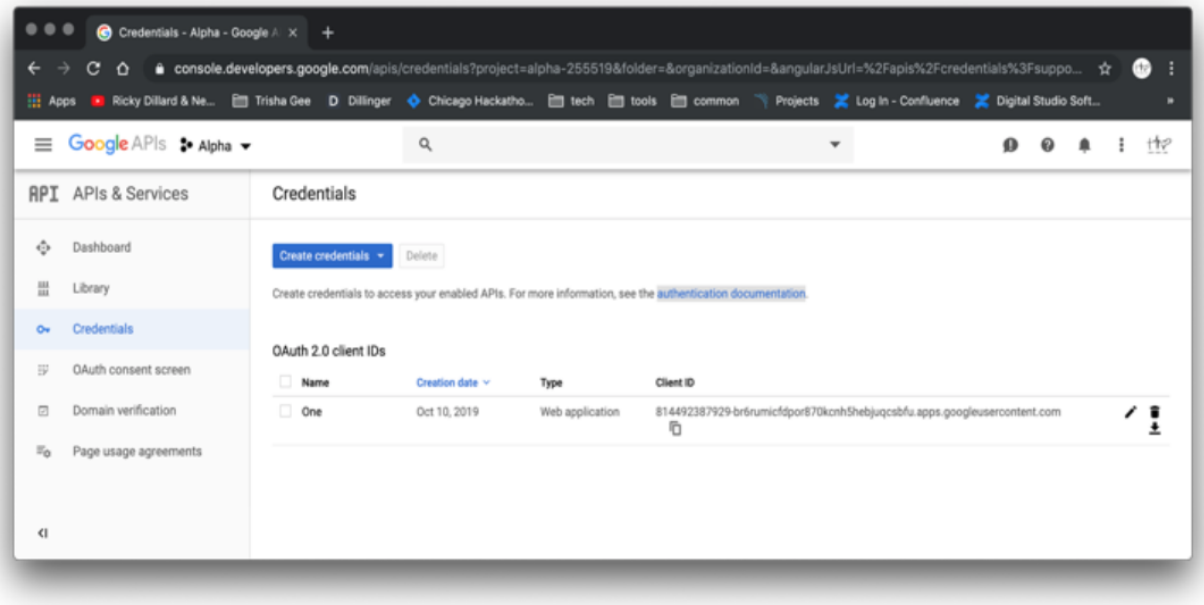
NB Because of the very nature of a browser, please clear cache for the browser on successive sessions "runs". This assures that you are accessing the OpenIDConnect-protected application in a way that consistently presents a modal login (challenge-response) dialog box.

Setup, Example

The screen shot below shows a completed Credential entry for a web application in the Google Developers API Console.

Notice that the value in the **Name** column becomes the title of the challenge/response login page.

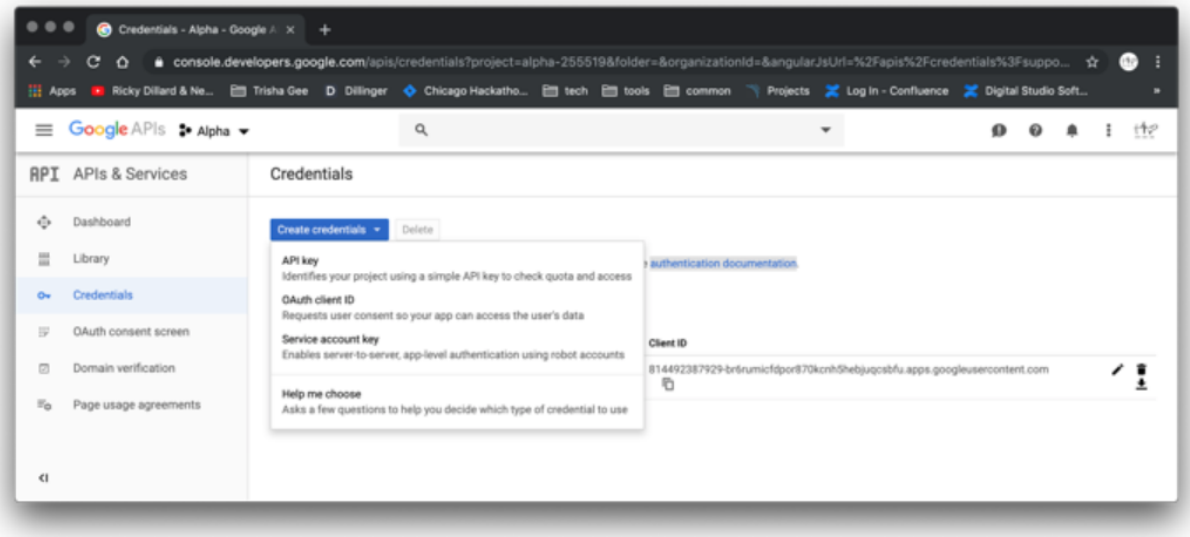
The value in the **Client ID** column below becomes the Key part of the K/V pair in the Spring's canonical `application.properties` file. This mapping, along with four other mappings, is shown later in this doc.



Setup, Process

The section illustrates how to create a Credential in Google's adaptation for OpenIDConnect OAuth2.

Below, choose the **OAuth client ID** option from the Create credentials pick list.

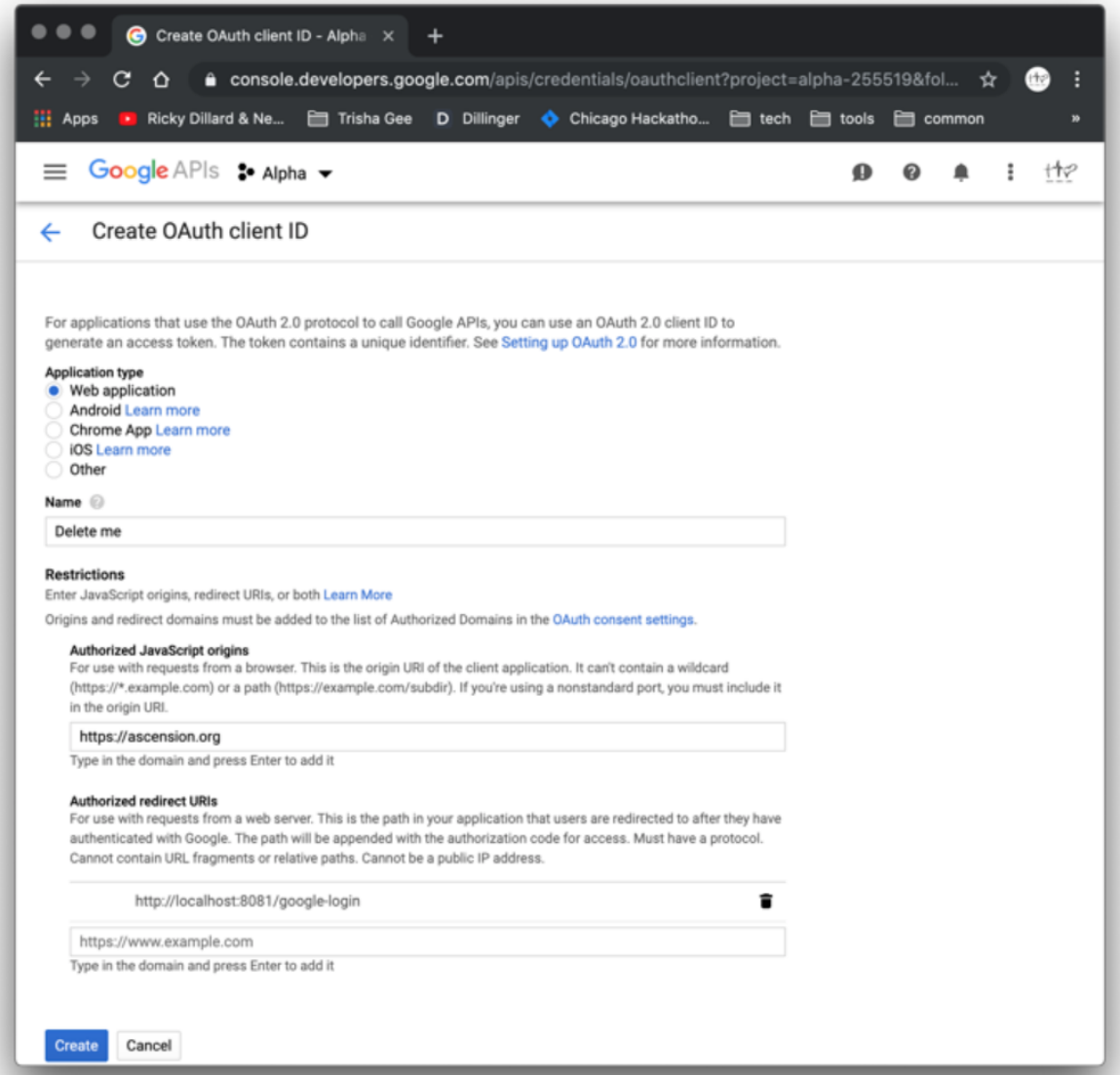


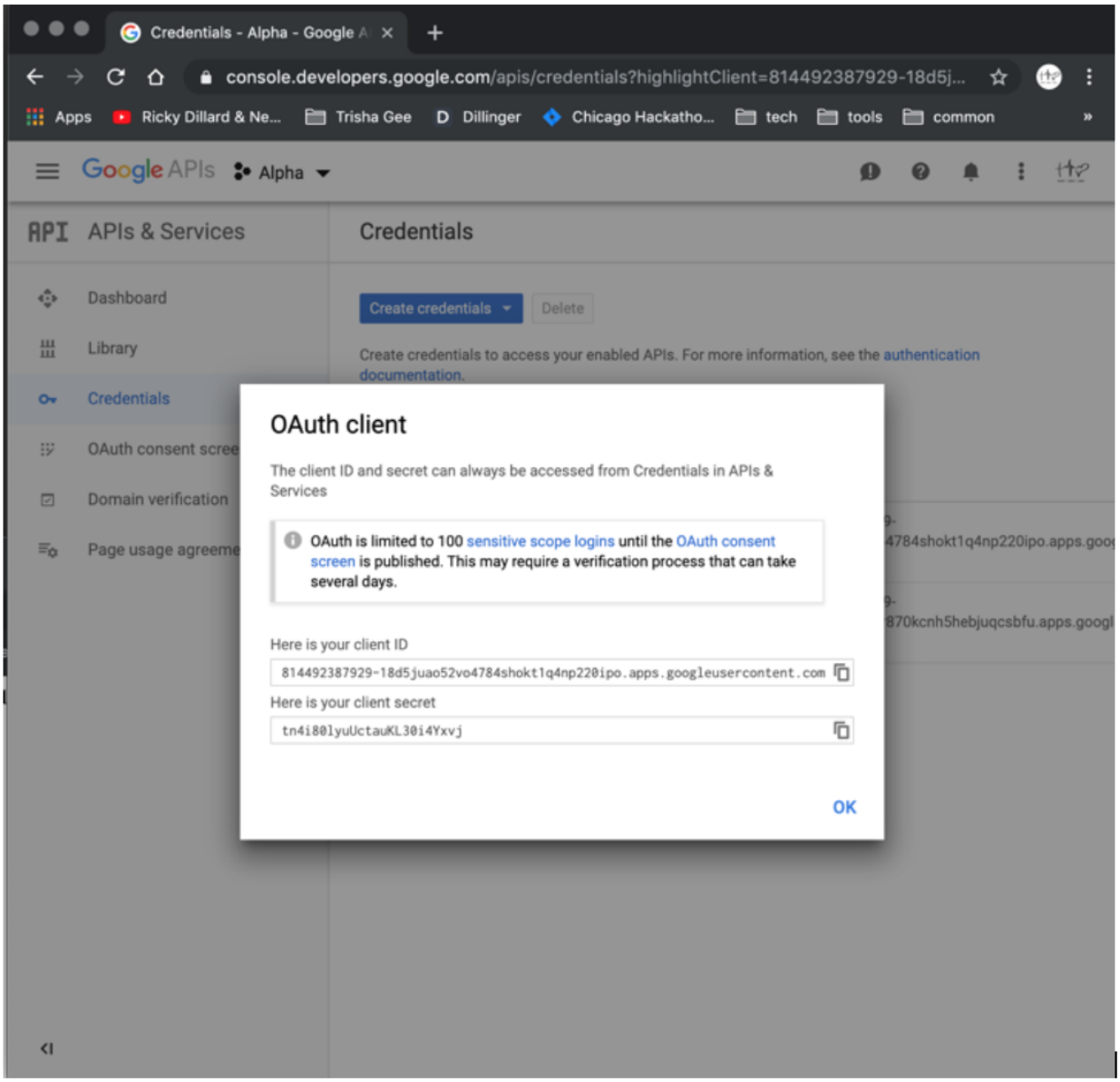
Below, choose the **Web application** radio button (option) in the **Application type** section.

Then, supply this value for the **Authorized redirect URIs** textbox:

`http://localhost:8081/google-login`

Click the Create button.

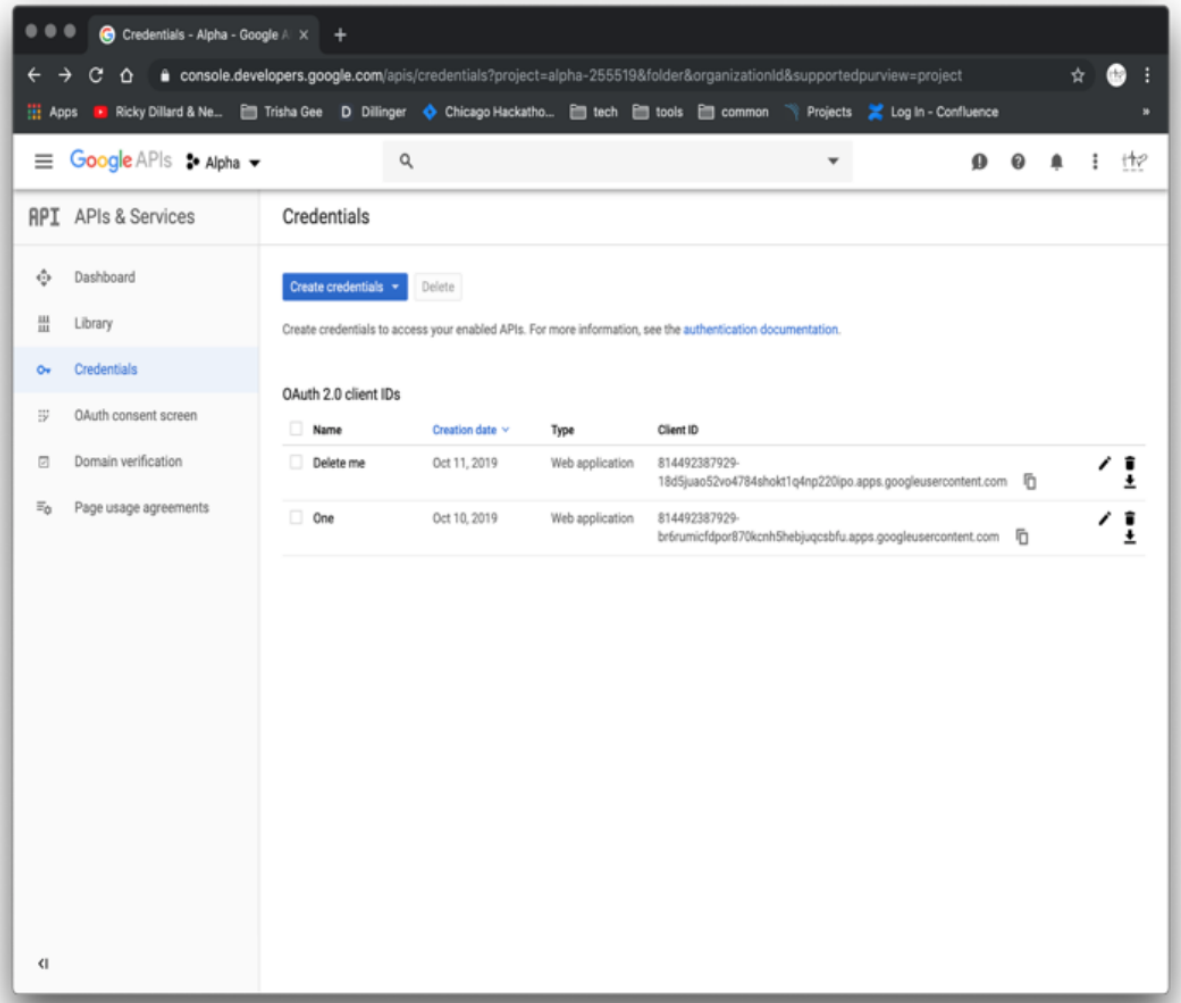




The pair of values generated by the Google's OpenIDConnect OAuth2 product are used as entries in Spring's canonical `application.properties` file.

They are the Client ID and the Client Secret.

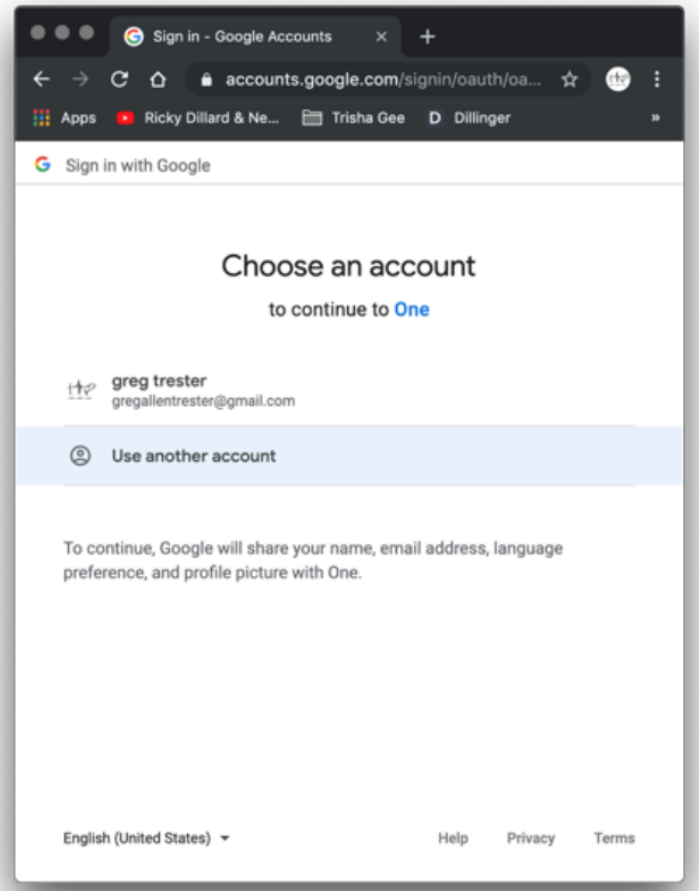
The screen shot below shows a second, completed Credential entry in the Google Developers API Console, for the web application that we just registered.

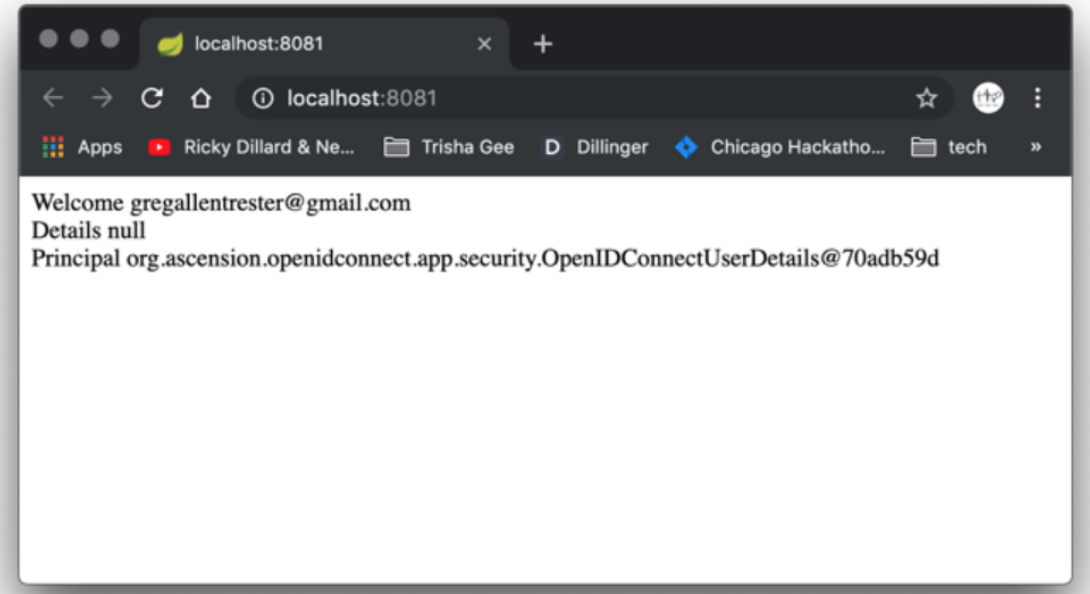


Usage

The browser URL/link for ingress into the OpenIDConnect-protected demo app is (by configurable design):

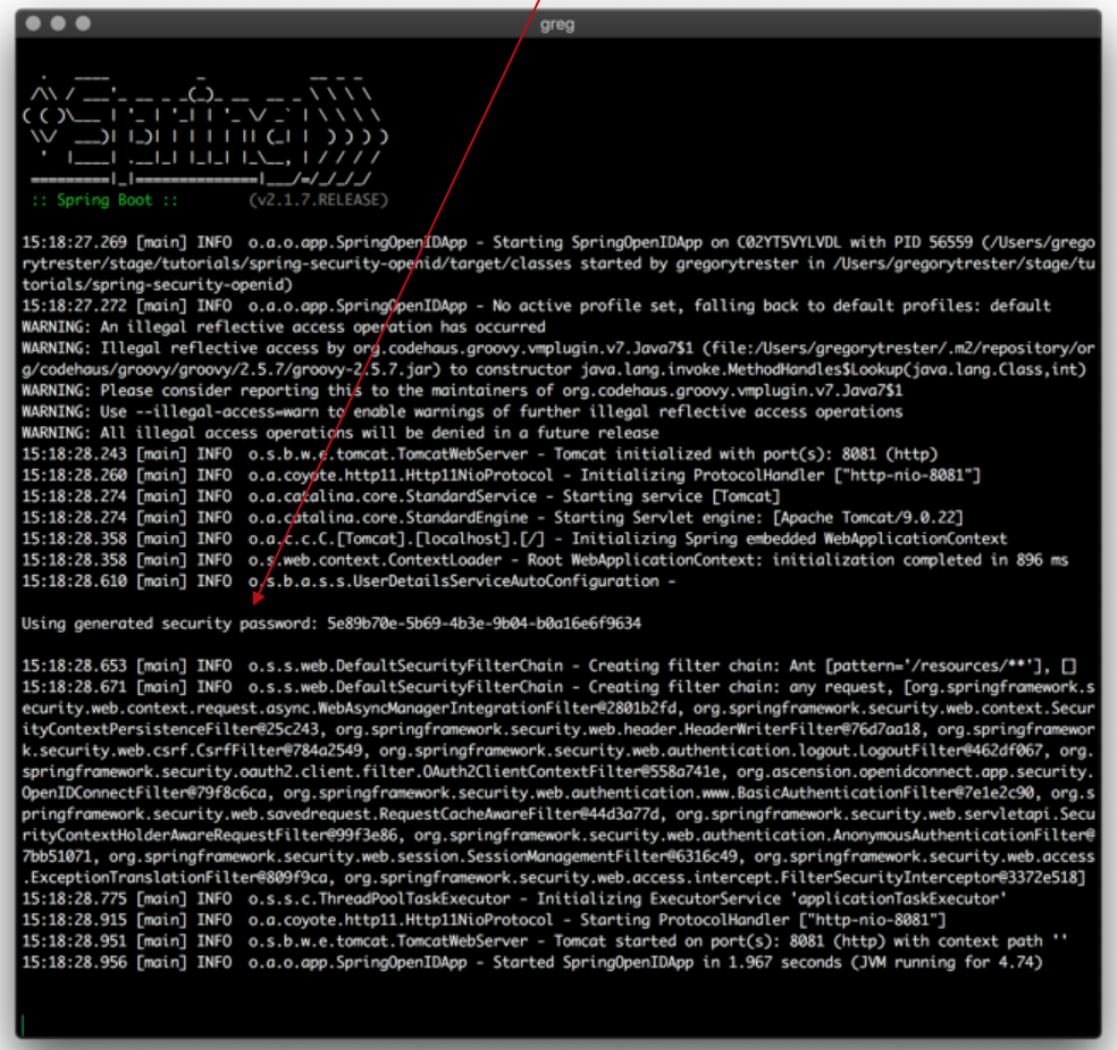
http://localhost:8081/google-login




localhost:8081
localhost:8081
Apps
Ricky Dillard & Ne...
Trisha Gee
Dillinger
Chicago Hackatho...
tech
Welcome gregallentrester@gmail.com
Details null
Principal org.ascension.openidconnect.app.security.OpenIDConnectUserDetails@70adb59d

The applications' console readout shows that a secure password has been generated (at the application/session scope); ACLs are not a part of the constraints being applied (for that, adopt the concept of a Principal).

The concept of a Principal provides an affordance of a standardized way to enforce a finer granularity of access to application assets (View/Update/Delete).

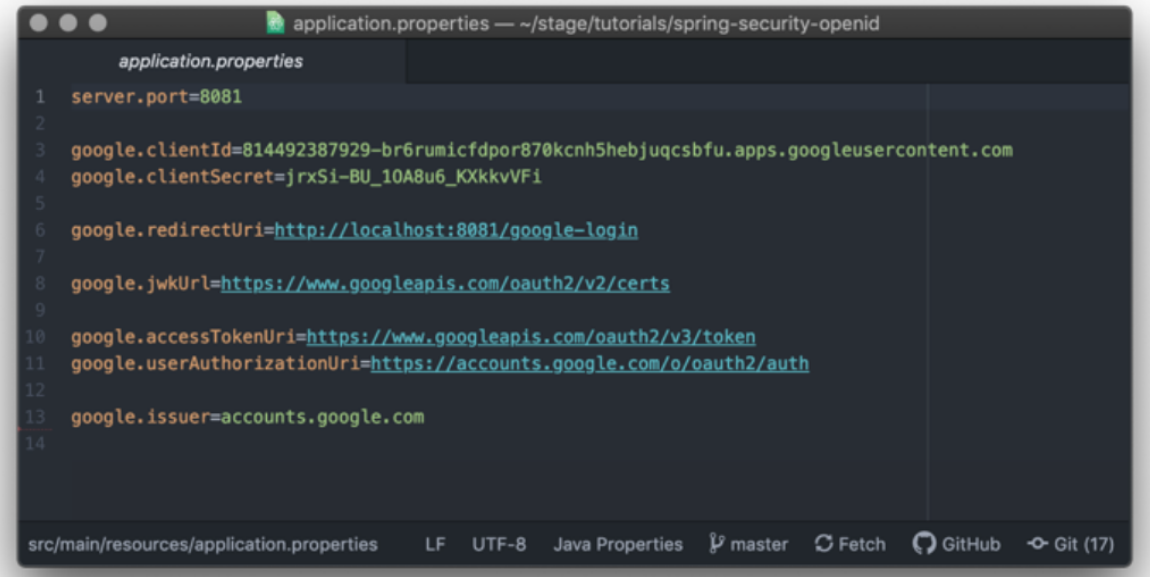


The Code

A Java application's part of the contract with OpenIDConnect is relatively unobtrusive.

The Spring Boot demo application that is discussed in this document uses a few annotations and some (5) entries in the canonical `application.properties` file.

NB In production, the `application.properties` file should have "locked-down" read only file permissions (e.g. `400`).



Notice above, that the values for the `clientId` and the `clientSecret` keys (lines 3 & 4, respectively) must match the values that were auto generated in the Credential section of the OpenID Console.

ßuild-Run

If you happen to have not stopped the Spring-boot application gracefully, the script show next will identify and kill your Spring-boot application.

Then, it compiles and executes the app by simply "sourcing" the script (w/o an extension, named:

```
ok
```

After killing the errant Spring-boot app, the `ok` script makes a delegating call to:

```
mvn spring-boot:run
```

ok - spring-boot:run

```
1  #!/bin/bash -p
2
3  ANY="spring"
4
5  ps auxx | grep "$ANY" | grep -v grep > "$ANY"
6
7  while IFS=' ' read -r line || [[ -n "$line" ]]; do
8
9    PID="$(echo $line | cut -d " " -f 2)"
10
11   sudo kill -9 "${PID}"
12
13 done < "$ANY"
14
15
16 mvn spring-boot:run -B

~/stage/tutorials/spring-security-openid
```

BTW, you can modify Line 3 in the above script to point to a fragment-of-a-name of any runtime image/process (ps, or process status).

Invocation

The browser URL/link for ingress into the OpenIDConnect-protected demo app is (by configurable design):

http://localhost:8081/google-login

Addendum

These links originate with Google; however, they represent best-practices for the OAuth2 topic.

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

https://developers.google.com/identity/protocols/OAuth2?hl=en_US

https://openid.net/connect/

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

https://github.com/eugenp/tutorials/tree/master/spring-security-openid

https://console.developers.google.com/apis/credentials?project=firstproject-98293&folder&organizationId

https://console.developers.google.com/apis/dashboard?project=firstproject-98293

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