OpenID Connect authentication

使用 Java 及 Nimbus OAuth 2.0 SDK with OpenID Connect extensions

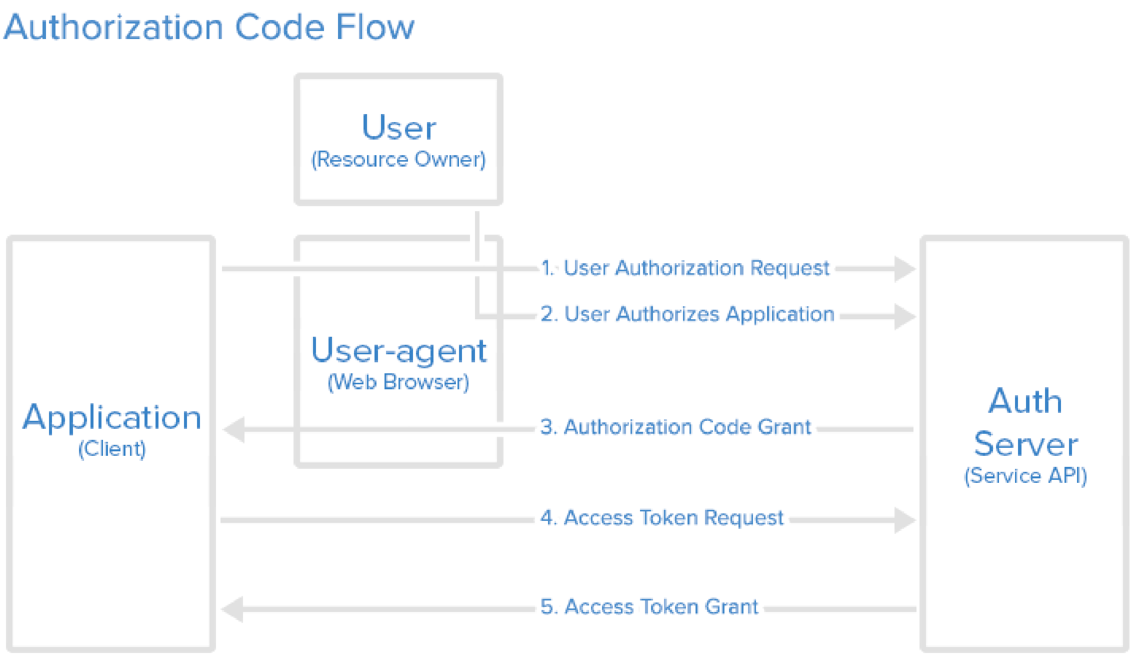
第一章 使用 OIDC 介接教育部帳號服務

demoApp 是一家前景看好,未來可能擁有數十萬客戶的網頁應用程式公司

https://coding.teliclab.info/demoApp/

本文件旨在協助建立 demoApp 透過 OIDC 與教育部帳號服務介接

採用模式如下:



參考: http://openid.net/specs/openid-connect-core-1_0.html#CodeFlowSteps

PREPARE:

向 Auth Server 申請 Client ID, 並且填寫 redicrect URI,

例如: demoApp 的首頁在 https://coding.teliclab.info/demoApp/

而我向 Auth Server 註冊的 redirect URI 為

https://coding.teliclab.info/demoApp/callback

註冊申請後, 會得到 Client ID 及 Client Secret, 依以下說明 http://n.sfs.tw/content/index/11209 為簡化日後管理及增加彈性, 建議使用 Discovery Document 來取得 authorization, token, userinfo 各 Endpoint

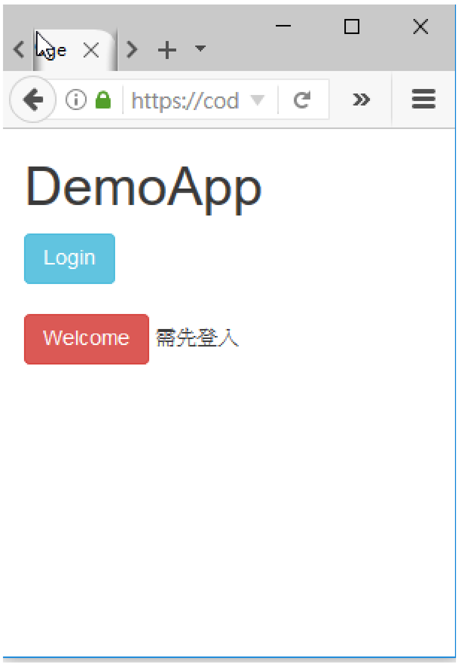
但本文件為了簡化程式碼及方便說明, 各 endpoint 採靜態值設定

本專案使用 maven 管理及需要的 repository

pom.xml

```
<dependencies>
      <dependency>
          <groupId>com.nimbusds</groupId>
          <artifactId>oauth2-oidc-sdk</artifactId>
          <version>5.27</version>
          <type>jar</type>
      </dependency>

     <dependency>
          <groupId>ch.qos.logback</groupId>
          <artifactId>logback-classic</artifactId>
          <version>1.2.3</version>
      </dependency>
      <dependency>
          <groupId>ch.qos.logback</groupId>
          <artifactId>logback-core</artifactId>
          <version>1.2.3</version>
      </dependency>

      <dependency>
          <groupId>javax</groupId>
          <artifactId>javaee-web-api</artifactId>
          <version>7.0</version>
          <scope>provided</scope>
      </dependency>
      <dependency>
          <groupId>org.slf4j</groupId>
          <artifactId>slf4j-api</artifactId>
          <version>1.7.25</version>
      </dependency>
</dependencies>
```

dempApp 的首頁,畫面如下



Welcome 按鈕連結的網頁帶有個人重要資訊屬不公開資訊, 因此必須先透過登入程序（即本文所做的 OIDC 介接）,才能看到網頁內容

1. User authorization request

點擊 login 按鈕將頁面導向 https://coding.teliclab.info/demoApp/login

此頁面的主要工作是為了在 user-agent 送出以下字串, 注意 參數值必須是 url encoding, 以 GET Request 向 Authorization Endpoint 傳送, 底下為一 sample url

https://oidc.tanet.edu.tw/oidc/v1/azp?response_type=code&client_id=CLIENTID&redirect_uri=https%3A%2F%2Fcoding.teliclab.info%2FdemoApp%2Fcallback&scope=openid+profile&state=STATE&nonce=NONCE

Authorization Endpoint 為 https://oidc.tanet.edu.tw/oidc/v1/azp

參數詳見 http://n.sfs.tw/content/index/11204

clientid 由 op 提供

scope 定義 access token 可以存取的權限 openid(必要), email(選項), profile(選項)

response_type code

redirect_uri 需申請時設定

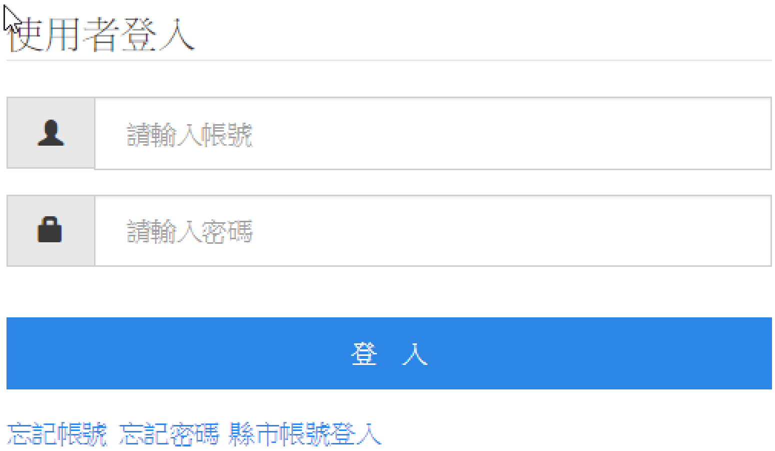
state 必要，RP 需由帶回來的值驗證 SESSION 連續狀態

nonce 必要 ，包含在要求中的值 (由應用程式所產生)，將會包含在所得的 `id_token` 中來做為宣告。應用程式接著便可確認此值，以減少權杖重新執行攻擊。

Login.java

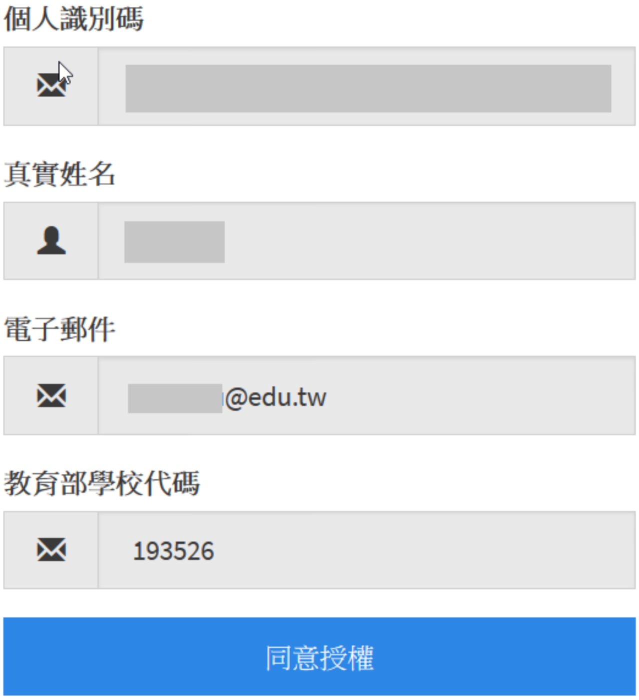
```
@WebServlet(name = "Login", urlPatterns = {"/login"})
public class Login extends HttpServlet {
    private final Logger logger =
LoggerFactory.getLogger(Login.class);

      protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
          throws ServletException, IOException,
SerializeException {
        HttpSession session = request.getSession();

        //The client identifier provisioned by the server
        ClientID clientID = new ClientID("CLIENID");

        // The client callback URI, pre-registered with the
server
        URI callback = new
URI("https://coding.teliclab.info/demoApp/callback");

    // Generate random state string for pairing the response to
the request 產生state值並存入session,與回應回傳的值做比對是否一致
         State state = new State();
         session.setAttribute("state", state.toString());

     // Generate nonce
          Nonce nonce = new Nonce();
     // Compose the request (in code flow)  如前面所提,組合字串並以GET
request

          AuthenticationRequest authzReq = new
AuthenticationRequest(
                new URI("https://oidc.tanet.edu.tw/oidc/v1/azp"),
                 new ResponseType("code"),
                 Scope.parse("openid profile"),
                 clientID,
                 callback,
                 state,
                 nonce);

         //此為 Authrozation Code flow 的第一步
         logger.info("1.User authorization request");
         //觀察送出的字串
          logger.info(authzReq.getEndpointURI().toString() + "?"
+ authzReq.toQueryString());
         //送出並導向

response.sendRedirect(authzReq.getEndpointURI().toString() + "?" +
authzReq.toQueryString());


       }
}
```

2.User authorize application


使用者登入

請輸入帳號

請輸入密碼


個人識別碼

[redacted]

真實姓名

[redacted]

電子郵件

[redacted]@edu.tw

教育部學校代碼

193526

3. Authorization Code Grant

前一個步驟使用者授權並且成功驗証後, Auth Server 會回傳一串參數值到我們所設定的 rediect URI, 所以這裡要取出 Authorization code 並且比對 state 值是否一致

Callback.java

```
String queryString = request.getQueryString();
String responseURL = "https:///path/?" + queryString;
//觀察回傳的值
logger.info(responseURL);

//取出先前建立的state 值, 稍後比對
HttpSession session = request.getSession();
String state = session.getAttribute("state").toString();
```

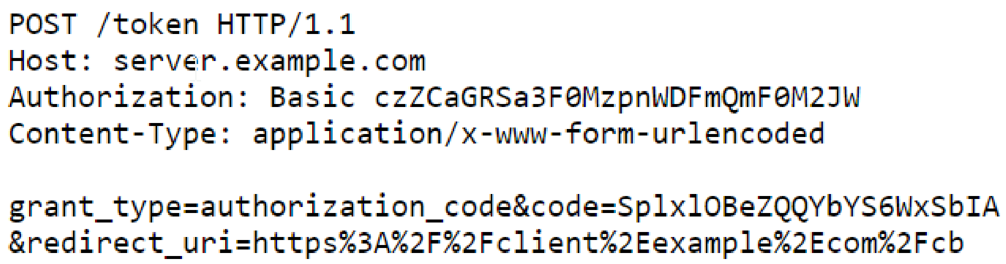
```
AuthenticationResponse authResponse =
  AuthenticationResponseParser.parse(new URI(responseURL));
AuthenticationSuccessResponse successResponse =
  (AuthenticationSuccessResponse) authResponse;

// 成功取得authorization code
AuthorizationCode code = successResponse.getAuthorizationCode();
logger.info("3. authz code grant.");
logger.info("code:" + code.toString());

logger.info("return state:" +
successResponse.getState().toString());

//比對state 的值是否一致
assert successResponse.getState().toString().equals(state);
```

4. Access Token Request

取到 Authorization Code 後, 利用此 code 跟 auth server 要 access token,規範如下

1. POST 方式傳送

2.參數值必須以 application/x-www-form-urlencoded 格式



```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
```

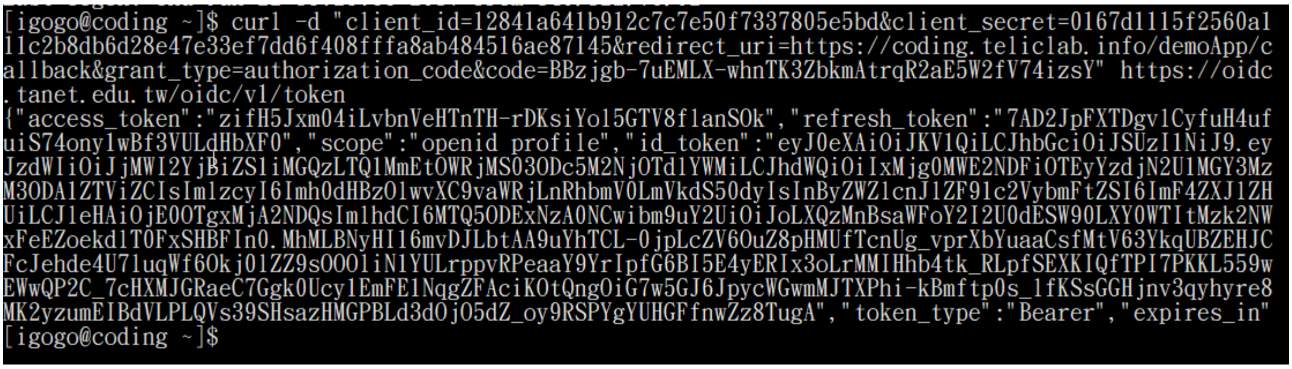
在開發階段, 為了方便除錯, 可利用 curl 指令觀察取得 access token 的結果

curl -d "client_id=CLIENTID&client_secret=CLIENTSECRET&redirect_uri=https://coding.teliclab.info/demoApp/callback&grant_type=authorization_code&code=AUTHZCODE" https://oidc.tanet.edu.tw/oidc/v1/token

-d 的參數代表以 POST 的方式傳送

上列整令為完整一行, 因版面關係產生斷行,參數部份以" " 雙引號包圍

實際操作圖例



```
[igogo@coding ~]$ curl -d "client_id=12841a641b912c7c7e50f7337805e5bd&client_secret=0167d1115f2560a1
11c2b8db6d28e47e33ef7dd6f408fffa8ab484516ae87145&redirect_uri=https://coding.teliclab.info/demoApp/c
allback&grant_type=authorization_code&code=BBzjgb-7uEMLX-whnTK3ZbkmAtrqR2aE5W2fV74izsY" https://oidc
.tanet.edu.tw/oidc/v1/token
{"access_token":"zifH5Jxm04iLvbnVeHTnTH-rDKsiYo15GTV8flanSOk","refresh_token":"7AD2JpFXTDgv1CyfuH4uf
uiS74ony1wBf3VULdHbXF0","scope":"openid profile","id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.ey
JzdWIiOiJjMWI2YjBiZS1iMGQzLTQ1MmEtOWRjMS03ODc5M2NjOTd1YWMiLCJhdWQiOiIxMjg0MWE2NDFiOTEyYzdjN2U1MGY3Mz
M3ODA1ZTViZCIsIm1zcyI6Imh0dHBzOlwvXC9vaWRjLnRhbmV0LmVkdS50dyIsInByZWZ1cnJ1ZF91c2VybmFtZSI6ImF4ZXJ1ZH
UiLCJ1eHAiOjE0OTgxMjA2NDQsIm1hdCI6MTQ5ODExNzA0NCwibm9uY2UiOiJoLXQzMnBsaWFoY2I2U0dESW90LXY0WTItMzk2NW
xFeEZoekd1T0FxSHBFIn0.MhMLBNyHI16mvDJLbtAA9uYhTCL-0jpLcZV6OuZ8pHMUfTcnUg_vprXbYuaaCsfMtV63YkqUBZEHJC
FcJehde4U71uqWf6Okj01ZZ9sOO01iN1YULrppvRPeaaY9YrIpfG6BI5E4yERIx3oLrMMIHhb4tk_RLpfSEXKIQfTPI7PKKL559w
EWwQP2C_7cHXMJGRaeC7Ggk0Ucy1EmFE1NqgZFAciKOtQng0iG7w5GJ6JpycWGwmMJTXPhi-kBmftp0s_1fKSsGGHjnv3qyhyre8
MK2yzumEIBdVLPLQVs39SHsazHMGPBLd3d0j05dZ_oy9RSPYgYUHGFfnwZz8TugA","token_type":"Bearer","expires_in"
[igogo@coding ~]$
```

Callback.java

```
//此處的code是接續前面自Authz endpoint 回應的值
AuthorizationCode code = ….
URI callback = new
URI("https://coding.teliclab.info/demoApp/callback");
AuthorizationGrant codeGrant = new AuthorizationCodeGrant(code,
callback);

//provisioned by the server
ClientID clientID = new ClientID("CLIENTID");
Secret clientSecret = new Secret("SECRET");
ClientAuthentication clientAuth = new ClientSecretPost(clientID,
clientSecret);

// The token endpoint
URI tokenEndpoint = new
URI("https://oidc.tanet.edu.tw/oidc/v1/token");

// Make the token request
TokenRequest tokenRequest= new TokenRequest(tokenEndpoint,
clientAuth, codeGrant);

 logger.info("4. Access Token Request");
 TokenResponse tokenResponse =
OIDCTokenResponseParser.parse(tokenRequest.toHTTPRequest().send())
;

//錯誤處理
if (tokenResponse instanceof TokenErrorResponse) {
    Correspondent errorResponse = (Correspondent) tokenResponse;
     logger.info(errorResponse.getErrorObject().getCode());
}
```

```
//使用OIDCTokenResponse 才能取得IDToken
//如果使用AccesstokenResponse只能取得Access Token
OIDCTokenResponse accessTokenResponse = (OIDCTokenResponse)
tokenResponse;
BearerAccessToken accessToken
                     =
accessTokenResponse.getOIDCTokens().getBearerAccessToken();

SignedJWT idToken =
      (SignedJWT)
accessTokenResponse.getOIDCTokens().getIDToken();

RefreshToken refreshToken = (RefreshToken)
accessTokenResponse.getOIDCTokens().getRefreshToken();

logger.info("5 Access Token Grant.");
logger.info("access token value:" + accessToken.getValue());
session.setAttribute("accessToken", accessToken.getValue());
```

```
//至此成功取得access token, Authorization Code flow流程跑完,將access
token 存入session,網頁導向welcome
response.sendRedirect("welcome");
```

使用 Access Token

取得合法的 access token 後, 我們可以利用此合法 access token 進行 API call, 例如取得 userinfo 的資訊，這裡我們透過三种方法中最安全也最推薦的 HTTP authorization header 傳送

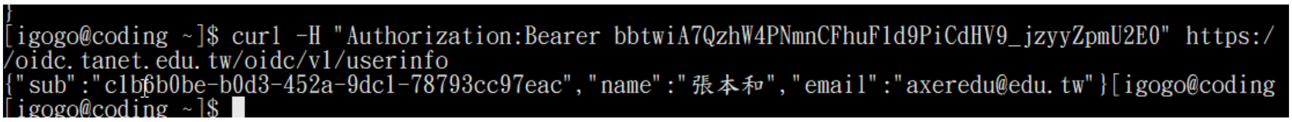
根據定義,詳見 https://tools.ietf.org/html/rfc6750 2.1

傳送 access token,我們還必須增加 Authorization header 到 GET request, 類似像這樣

```
GET /resource HTTP/1.1
Host: server.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

利用 curl 指令可以更改 request headers 作為模擬請求

curl -H "Authorization:Bearer ACCESS TOKEN VALUE" https://oidc.tanet.edu.tw/oidc/v1/userinfo

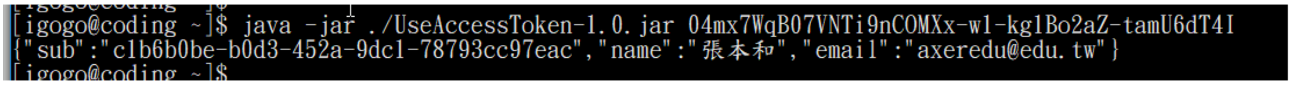


```
}
[igogo@coding ~]$ curl -H "Authorization:Bearer bbtwiA7QzhW4PNmnCFhuF1d9PiCdHV9_jzyyZpmU2E0" https:/
/oidc.tanet.edu.tw/oidc/v1/userinfo
{"sub":"c1b6b0be-b0d3-452a-9dc1-78793cc97eac","name":"張本和","email":"axeredu@edu.tw"}[igogo@coding
[igogo@coding ~]$ 
```

底下程式片段並不屬於demoApp的部份, 此段旨在說明如何增加Authorization Header, 我們可使用Apache HTTP Client library,請注意如果使用nimbus library,則直接以BearerAccessToken對API 進行request即可

```
package app.demo.useaccesstoken;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;

public class Main {
    public static void main(String[] args) throws
URISyntaxException, UnsupportedEncodingException, IOException {

        if (args.length > 0) {
          //取得access token後 以參數帶入的方式執行本程式
            String accessToken = args[0];
            URI userinfoEndpointURL = new
URI("https://oidc.tanet.edu.tw/oidc/v1/userinfo");
            CloseableHttpClient httpClient =
HttpClients.createDefault();
            HttpPost httpPost = new HttpPost(userinfoEndpointURL);
            httpPost.addHeader("Authorization", "Bearer "+
accessToken);
            List<NameValuePair> urlParameters = new ArrayList<>();

            urlParameters.add(new BasicNameValuePair("method","get"));
            httpPost.setEntity(new UrlEncodedFormEntity(urlParameters));

            HttpResponse httpResponse = httpClient.execute(httpPost);

            //Extract data from response
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(httpResponse.getEntity().getContent()));
            System.out.println(bufferedReader.readLine());
        }

    }

}
```

執行結果



```
[igogo@coding ~]$ java -jar ./UseAccessToken-1.0.jar 04mx7WqB07VNTi9nCOMXx-w1-kg1Bo2aZ-tamU6dT4I
{"sub":"c1b6b0be-b0d3-452a-9dc1-78793cc97eac","name":"張本和","email":"axeredu@edu.tw"}
[igogo@coding ~]$
```

demoApp 使用 nimbus library 請看 Welcome.java 的程式碼片段

Welcome.java

```
HttpSession session = request.getSession();
if (session.getAttribute("accessToken") == null) {
   //not allowed to visit this page
}else{
   //以 Bearer AccessToken  對 userinfo endpoint 進行 request
   BearerAccessToken accessToken = new
BearerAccessToken((session.getAttribute("accessToken").toString())
);

    URI userinfoEndpointURL = new
URI("https://oidc.tanet.edu.tw/oidc/v1/userinfo");
    // Append the access token to form actual request
    UserInfoRequest userInfoReq = new
UserInfoRequest(userinfoEndpointURL, accessToken);

   //觀察送出的 header
   logger.info("userinfo request header:"+
userInfoReq.toHTTPRequest().getHeaders().toString());


  userInfoHTTPResponse = userInfoReq.toHTTPRequest().send();
  userInfoResponse =
UserInfoResponse.parse(userInfoHTTPResponse);
  UserInfoSuccessResponse successUserInfoResponse =
(UserInfoSuccessResponse) userInfoResponse;
   String msg =
successUserInfoResponse.getUserInfo().toJSONObject().toString();
    //處理 msg 輸出到網頁...

}
```

驗証 ID TOKEN

詳見 http://n.sfs.tw/content/index/11231

```
// Set up a JWT processor to parse the tokens and then check their
signature
// and validity time window (bounded by the "iat", "nbf" and "exp"
claims) ConfigurableJWTProcessor jwtProcessor = new
DefaultJWTProcessor();

// The public RSA keys to validate the signatures will be sourced
from the
// OAuth 2.0 server's JWK set, published at a well-known URL. The
RemoteJWKSet
// object caches the retrieved keys to speed up subsequent look-
ups and can
// also gracefully handle key-rollover

JWKSource keySource = new RemoteJWKSet(new
URL("https://oidc.tanet.edu.tw/oidc/v1/jwksets"));
JWKSource keySource = new RemoteJWKSet(new URL(jwksURI));

// The expected JWS algorithm of the access tokens (agreed out-of-
band)
JWSAlgorithm expectedJWSAlg = JWSAlgorithm.RS256;

// Configure the JWT processor with a key selector to feed
matching public
// RSA keys sourced from the JWK set URL
JWSKeySelector keySelector = new
JWSVerificationKeySelector(expectedJWSAlg, keySource);
jwtProcessor.setJWSKeySelector(keySelector);
// Process the token
SecurityContext ctx = null; // optional context parameter, not
required here
JWTClaimsSet claimsSet = jwtProcessor.process(idToken, ctx);

String idTokenParsed = claimsSet.toString();
```

完整程式碼

Index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Start Page</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
        <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
    </head>
    <body>
        <div class="container">
            <h1>DemoApp</h1>

            <a href="login" class="btn btn-info" role="button">MOE Login </a>

            <br/>
            <br/>

            <a href="welcome" class="btn btn-danger" role="button">Welcome</a>  需先登入
        </div>
    </body>
</html>
```

Login.java

```
package app.demo.demoapp;

import com.nimbusds.oauth2.sdk.ResponseType;
import com.nimbusds.oauth2.sdk.Scope;
import com.nimbusds.oauth2.sdk.SerializeException;
import com.nimbusds.oauth2.sdk.id.ClientID;
import com.nimbusds.oauth2.sdk.id.State;
import com.nimbusds.openid.connect.sdk.AuthenticationRequest;
import com.nimbusds.openid.connect.sdk.Nonce;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.logging.Level;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


@WebServlet(name = "Login", urlPatterns = {"/login"})
public class Login extends HttpServlet {

    private final Logger logger = LoggerFactory.getLogger(Login.class);

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException, SerializeException {
        HttpSession session = request.getSession();

        try {

            // The client identifier provisioned by the server
            //MOE clientid
            ClientID clientID = new ClientID("CLIENTID");
            // The client callback URI, typically pre-registered with the server
            URI callback = new URI("https://coding.teliclab.info/demoApp/callback");
            // Generate random state string for pairing the response to the request
            State state = new State();
            session.setAttribute("state", state.toString());

            // Generate nonce
            Nonce nonce = new Nonce();

            // Compose the request (in code flow)
            AuthenticationRequest authzReq = new AuthenticationRequest(
                    new URI("https://oidc.tanet.edu.tw/oidc/v1/azp"),
                    new ResponseType("code"),
                    Scope.parse("openid profile"),
                    clientID,
                    callback,
                    state,
                    nonce);

            logger.info("1.User authorization request");

            logger.info(authzReq.getEndpointURI().toString() + "?" +
authzReq.toQueryString());
            response.sendRedirect(authzReq.getEndpointURI().toString() + "?" +
authzReq.toQueryString());

        } catch (URISyntaxException ex) {
            logger.info(ex.getMessage());
        }

    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SerializeException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null,
ex);
```

```
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SerializeException ex) {
            java.util.logging.Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

}
```

Welcome.java

```
package app.demo.demoapp;

import com.nimbusds.oauth2.sdk.ErrorObject;
import com.nimbusds.oauth2.sdk.ParseException;
import com.nimbusds.oauth2.sdk.SerializeException;
import com.nimbusds.oauth2.sdk.http.HTTPResponse;
import com.nimbusds.oauth2.sdk.token.BearerAccessToken;
import com.nimbusds.openid.connect.sdk.UserInfoErrorResponse;
import com.nimbusds.openid.connect.sdk.UserInfoRequest;
import com.nimbusds.openid.connect.sdk.UserInfoResponse;
import com.nimbusds.openid.connect.sdk.UserInfoSuccessResponse;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.util.logging.Level;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@WebServlet(name = "Welcome", urlPatterns = {"/welcome"})
public class Welcome extends HttpServlet {

    private final Logger logger = LoggerFactory.getLogger(Callback.class);
    URI userinfoEndpointURL;

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     * @throws java.net.URISyntaxException
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException, URISyntaxException {
        HttpSession session = request.getSession();
```

```
      if (session.getAttribute("accessToken") == null) {

         request.setAttribute("msg", "Not Allowed");

         RequestDispatcher dispatcher = request
               .getRequestDispatcher("forbidden.jsp");
         dispatcher.forward(request, response);

      } else {
         BearerAccessToken accessToken = new
BearerAccessToken((session.getAttribute("accessToken").toString()));

         //https://connect2id.com/products/nimbus-oauth-openid-connect-sdk/guides/java-
cookbook-for-openid-connect-public-clients

          userinfoEndpointURL = new URI("https://oidc.tanet.edu.tw/oidc/v1/userinfo");

         // Append the access token to form actual request
         UserInfoRequest userInfoReq = new UserInfoRequest(userinfoEndpointURL,
accessToken);
         HTTPResponse userInfoHTTPResp = null;
         try {
            userInfoHTTPResp = userInfoReq.toHTTPRequest().send();
         } catch (SerializeException | IOException e) {
            // TODO proper error handling
         }

         UserInfoResponse userInfoResponse = null;
         try {
            userInfoResponse = UserInfoResponse.parse(userInfoHTTPResp);
         } catch (ParseException e) {
            // TODO proper error handling
         }

         if (userInfoResponse instanceof UserInfoErrorResponse) {
            ErrorObject error = ((UserInfoErrorResponse)
userInfoResponse).getErrorObject();
            // TODO error handling
         }

         UserInfoSuccessResponse successUserInfoResponse =
(UserInfoSuccessResponse) userInfoResponse;
         String msg = successUserInfoResponse.getUserInfo().toJSONObject().toString();
         request.setAttribute("msg", msg);
         RequestDispatcher dispatcher = request
               .getRequestDispatcher("welcome.jsp");
         dispatcher.forward(request, response);

      }
```

```
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (URISyntaxException ex) {
            java.util.logging.Logger.getLogger(Welcome.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (URISyntaxException ex) {
            java.util.logging.Logger.getLogger(Welcome.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }

}
```

Callback.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package app.demo.demoapp;

import com.nimbusds.jwt.SignedJWT;
import com.nimbusds.oauth2.sdk.AuthorizationCode;
import com.nimbusds.oauth2.sdk.AuthorizationCodeGrant;
import com.nimbusds.oauth2.sdk.AuthorizationGrant;
import com.nimbusds.oauth2.sdk.SerializeException;
import com.nimbusds.oauth2.sdk.TokenRequest;
import com.nimbusds.oauth2.sdk.TokenResponse;
import com.nimbusds.oauth2.sdk.TokenErrorResponse;
import com.nimbusds.oauth2.sdk.auth.ClientAuthentication;
import com.nimbusds.oauth2.sdk.auth.ClientSecretPost;
import com.nimbusds.oauth2.sdk.auth.Secret;
import com.nimbusds.oauth2.sdk.id.ClientID;
import com.nimbusds.oauth2.sdk.token.BearerAccessToken;
import com.nimbusds.oauth2.sdk.token.RefreshToken;
import com.nimbusds.openid.connect.sdk.AuthenticationResponse;
import com.nimbusds.openid.connect.sdk.AuthenticationResponseParser;
import com.nimbusds.openid.connect.sdk.AuthenticationSuccessResponse;
import com.nimbusds.openid.connect.sdk.OIDCTokenResponse;
import com.nimbusds.openid.connect.sdk.OIDCTokenResponseParser;
import java.io.IOException;
import java.net.URI;
import java.util.logging.Level;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 *
 * @author igogo
 */
@WebServlet(name = "Callback", urlPatterns = {"/callback"})
public class Callback extends HttpServlet {

    private final Logger logger = LoggerFactory.getLogger(Callback.class);

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
    *
    * @param request servlet request
    * @param response servlet response
    * @throws ServletException if a servlet-specific error occurs
    * @throws IOException if an I/O error occurs
    * @throws com.nimbusds.oauth2.sdk.SerializeException
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
         throws ServletException, IOException, SerializeException {

      try {

         //logger.info(request.getQueryString());
         String queryString = request.getQueryString();
         String responseURL = "https:///path/?" + queryString;
         logger.info(responseURL);
         HttpSession session = request.getSession();
         logger.info("session state:" + session.getAttribute("state"));
         String state = session.getAttribute("state").toString();

         AuthenticationResponse authResponse =
AuthenticationResponseParser.parse(new URI(responseURL));
         AuthenticationSuccessResponse successResponse =
(AuthenticationSuccessResponse) authResponse;
         // Retrieve the authorisation code
         AuthorizationCode code = successResponse.getAuthorizationCode();
         logger.info("3. authz code grant.");
         logger.info("code:" + code.toString());

         logger.info("return state:" + successResponse.getState().toString());
         assert successResponse.getState().toString().equals(state);
         logger.info("the same state");

         URI callback = new URI("https://coding.teliclab.info/demoApp/callback");
         AuthorizationGrant codeGrant = new AuthorizationCodeGrant(code, callback);

         //provisioned by the server
         ClientID clientID = new ClientID("CLIENTID");
         Secret clientSecret = new Secret("SECRET");

         ClientAuthentication clientAuth = new ClientSecretPost(clientID, clientSecret);

         // The token endpoint
         URI tokenEndpoint = new URI("https://oidc.tanet.edu.tw/oidc/v1/token");
         // Make the token request
         TokenRequest tokenRequest
               = new TokenRequest(tokenEndpoint, clientAuth, codeGrant);

//             logger.info("Token Authorization Header : " + httpRequest.getAuthorization());
```

```
        logger.info("4. Access Token Request");
        TokenResponse tokenResponse =
OIDCTokenResponseParser.parse(tokenRequest.toHTTPRequest().send());
        if (tokenResponse instanceof TokenErrorResponse) {

          TokenErrorResponse errorResponse = (TokenErrorResponse) tokenResponse;
          logger.info(errorResponse.getErrorObject().getCode());
        } else {
          OIDCTokenResponse accessTokenResponse = (OIDCTokenResponse)
tokenResponse;
          BearerAccessToken accessToken
              = accessTokenResponse.getOIDCTokens().getBearerAccessToken();

          SignedJWT idToken = (SignedJWT)
accessTokenResponse.getOIDCTokens().getIDToken();
          RefreshToken refreshToken = (RefreshToken)
accessTokenResponse.getOIDCTokens().getRefreshToken();

          logger.info("5 Access Token Grant.");
          logger.info("access token value:" + accessToken.getValue());
          session.setAttribute("accessToken", accessToken.getValue());
          response.sendRedirect("welcome");
        }
      } catch (Exception ex) {
        logger.info(ex.getMessage());
      }

  }

  // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign
on the left to edit the code.">
  /**
   * Handles the HTTP <code>GET</code> method.
   *
   * @param request servlet request
   * @param response servlet response
   * @throws ServletException if a servlet-specific error occurs
   * @throws IOException if an I/O error occurs
   */
  @Override
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      try {
        processRequest(request, response);
      } catch (SerializeException ex) {
        java.util.logging.Logger.getLogger(Callback.class.getName()).log(Level.SEVERE,
null, ex);
      }
  }
```

```
    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SerializeException ex) {
            logger.debug(ex.getMessage());
        }
    }

}
```

Welcome.java

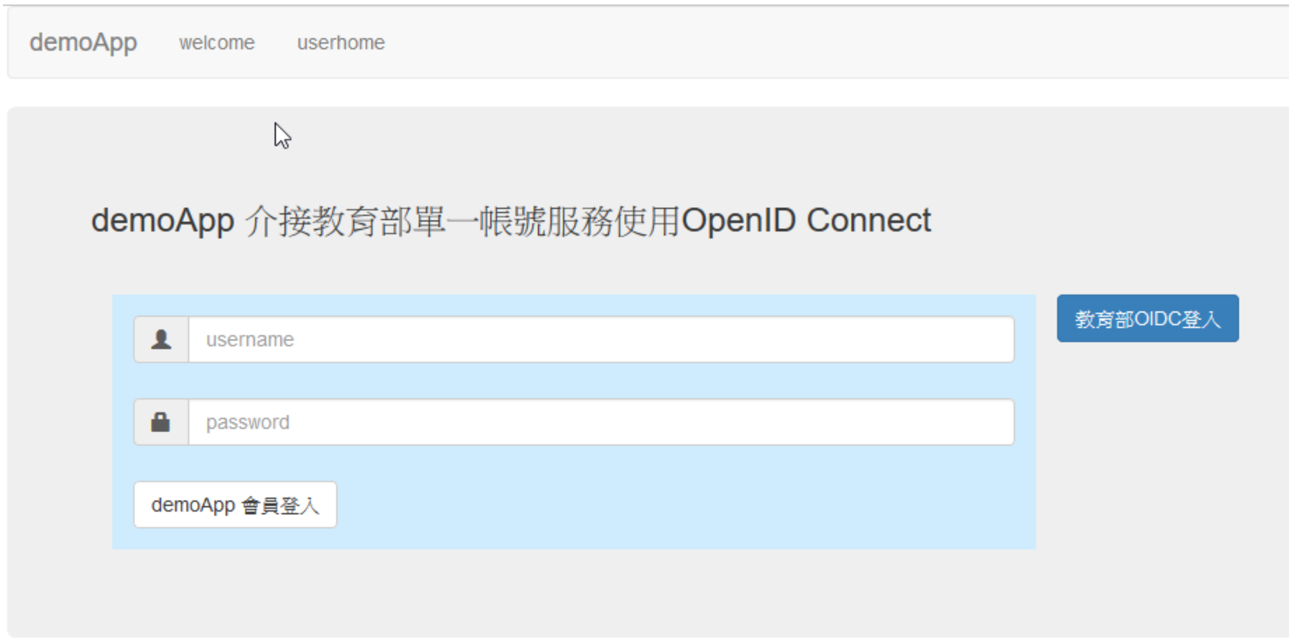
```
package app.demo.demoapp;

import com.nimbusds.jose.JOSEException;
import com.nimbusds.jose.JWSAlgorithm;
import com.nimbusds.jose.jwk.source.JWKSource;
import com.nimbusds.jose.jwk.source.RemoteJWKSet;
import com.nimbusds.jose.proc.BadJOSEException;
import com.nimbusds.jose.proc.JWSKeySelector;
import com.nimbusds.jose.proc.JWSVerificationKeySelector;
import com.nimbusds.jose.proc.SecurityContext;
import com.nimbusds.jwt.JWTClaimsSet;
import com.nimbusds.jwt.proc.ConfigurableJWTProcessor;
import com.nimbusds.jwt.proc.DefaultJWTProcessor;
import com.nimbusds.oauth2.sdk.ErrorObject;
import com.nimbusds.oauth2.sdk.ParseException;
import com.nimbusds.oauth2.sdk.SerializeException;
import com.nimbusds.oauth2.sdk.http.HTTPResponse;
import com.nimbusds.oauth2.sdk.token.BearerAccessToken;
import com.nimbusds.openid.connect.sdk.UserInfoErrorResponse;
import com.nimbusds.openid.connect.sdk.UserInfoRequest;
import com.nimbusds.openid.connect.sdk.UserInfoResponse;
import com.nimbusds.openid.connect.sdk.UserInfoSuccessResponse;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.logging.Level;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@WebServlet(name = "Welcome", urlPatterns = {"/welcome"})
public class Welcome extends HttpServlet {

    private final Logger logger = LoggerFactory.getLogger(Callback.class);

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
    */
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException, URISyntaxException {
      HttpSession session = request.getSession();
      if (session.getAttribute("accessToken") == null) {

         request.setAttribute("msg", "Not Allowed");

         RequestDispatcher dispatcher = request
               .getRequestDispatcher("forbidden.jsp");
         dispatcher.forward(request, response);

      } else {
         BearerAccessToken accessToken = new
BearerAccessToken((session.getAttribute("accessToken").toString()));

//               https://connect2id.com/products/nimbus-oauth-openid-connect-sdk/guides/java-
cookbook-for-openid-connect-public-clients
         URI userinfoEndpointURL = new URI("https://oidc.tanet.edu.tw/oidc/v1/userinfo");
         // Append the access token to form actual request
         UserInfoRequest userInfoReq = new UserInfoRequest(userinfoEndpointURL,
accessToken);
         HTTPResponse userInfoHTTPResponse = null;
         try {
            userInfoHTTPResponse = userInfoReq.toHTTPRequest().send();
         } catch (SerializeException | IOException e) {
            // TODO proper error handling
         }

         UserInfoResponse userInfoResponse = null;
         try {
            userInfoResponse = UserInfoResponse.parse(userInfoHTTPResponse);
         } catch (ParseException e) {
            // TODO proper error handling
         }

         if (userInfoResponse instanceof UserInfoErrorResponse) {
            ErrorObject error = ((UserInfoErrorResponse) userInfoResponse).getErrorObject();
            // TODO error handling
         }

         UserInfoSuccessResponse successUserInfoResponse = (UserInfoSuccessResponse)
userInfoResponse;
         String msg = successUserInfoResponse.getUserInfo().toJSONObject().toString();
         request.setAttribute("msg", msg);

         // Set up a JWT processor to parse the tokens and then check their signature
         // and validity time window (bounded by the "iat", "nbf" and "exp" claims)
         ConfigurableJWTProcessor jwtProcessor = new DefaultJWTProcessor();
```

```
            // The public RSA keys to validate the signatures will be sourced from the
            // OAuth 2.0 server's JWK set, published at a well-known URL. The RemoteJWKSet
           // object caches the retrieved keys to speed up subsequent look-ups and can
           // also gracefully handle key-rollover

         JWKSource keySource = new RemoteJWKSet(new
URL("https://oidc.tanet.edu.tw/oidc/v1/jwksets"));



            // The expected JWS algorithm of the access tokens (agreed out-of-band)
            JWSAlgorithm expectedJWSAlg = JWSAlgorithm.RS256;

            // Configure the JWT processor with a key selector to feed matching public
            // RSA keys sourced from the JWK set URL
            JWSKeySelector keySelector = new JWSVerificationKeySelector(expectedJWSAlg,
keySource);
            jwtProcessor.setJWSKeySelector(keySelector);

            // Process the token
            SecurityContext ctx = null; // optional context parameter, not required here
            JWTClaimsSet claimsSet = jwtProcessor.process(idToken, ctx);

            logger.info(claimsSet.toString());



            RequestDispatcher dispatcher = request
                    .getRequestDispatcher("welcome.jsp");
            dispatcher.forward(request, response);

        }

    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the
left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (URISyntaxException ex) {
            java.util.logging.Logger.getLogger(Welcome.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (URISyntaxException ex) {
            java.util.logging.Logger.getLogger(Welcome.class.getName()).log(Level.SEVERE, null, 
ex);
        }
    }

}
```

第二章 與現有系統帳號綁定

本章旨在說明在與教育部帳號服務介接後, 如何與現存資料庫中的帳號進行綁定, 若是由 openid2.0 的服務遷移到 OIDC, 處理原則是一樣的, 請在 scope 中加上 openid2, 由取得的 id token 中拿到 open2_id 的值再進行邏輯判斷

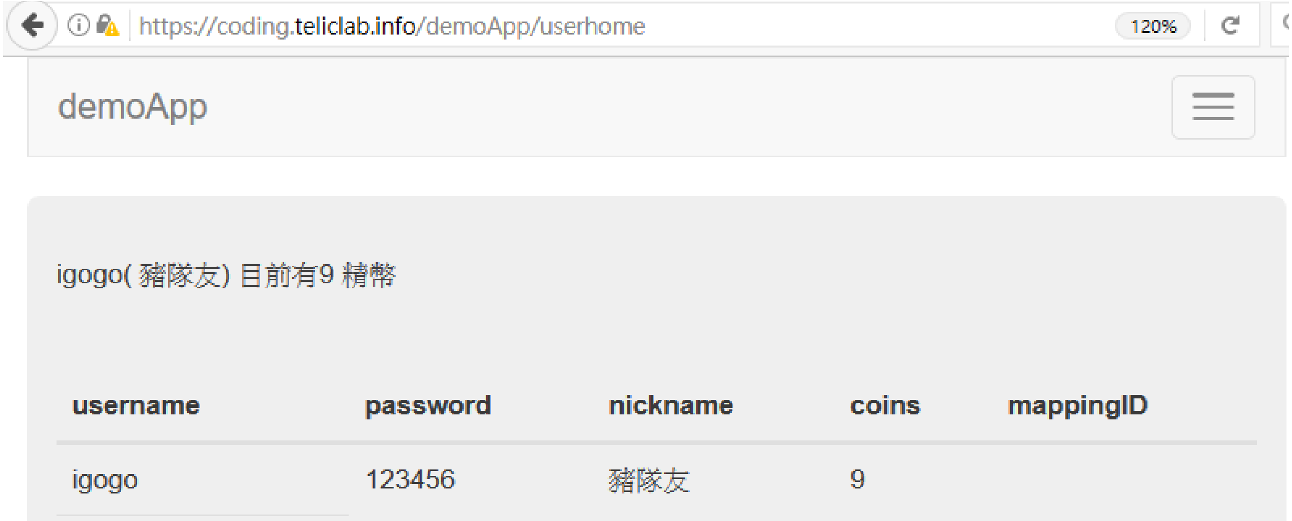
demoApp 原先已擁有會員帳號,在成功介接教育部帳號服務後,想與資料庫中的帳號綁定,以記錄原先帳號的使用歷程及資料, 如下圖示



demoApp 介接教育部單一帳號服務使用OpenID Connect

welcome 是延續上章在成功界接 OP 後的頁面, 成功登入會得到 userinfo 的值

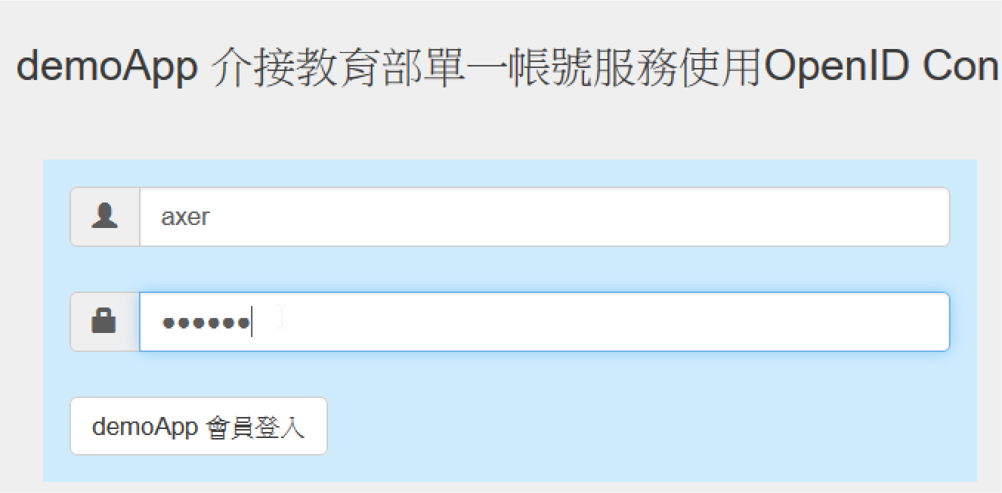
userhome 則為原 dempApp 會員登入後的使用者頁面, 成功登入後會取得使用者帳號的資料及使用歷程



igogo(豬隊友) 目前有9 精幣

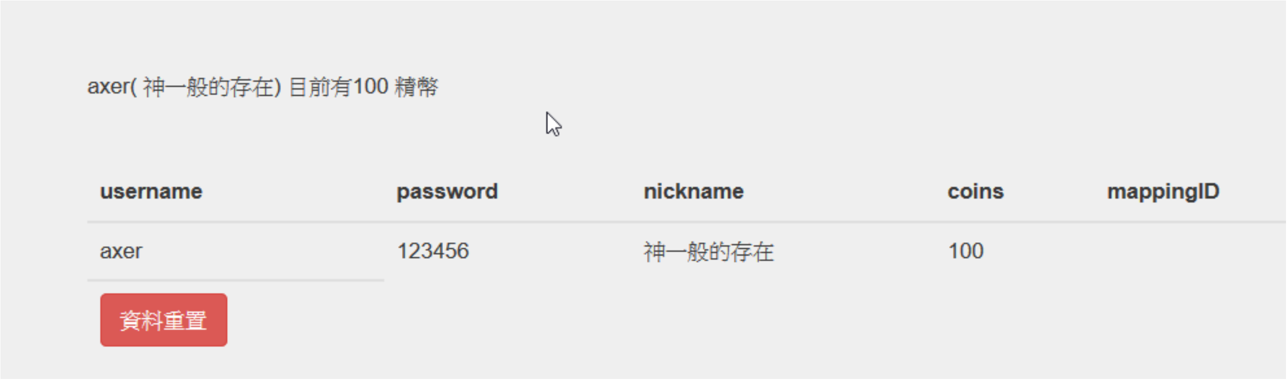
username	password	nickname	coins	mappingID
igogo	123456	豬隊友	9	

以 dempApp 資料庫帳號的登入流程示意如下：



demoApp 介接教育部單一帳號服務使用OpenID Con

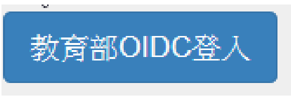
登入後的畫面

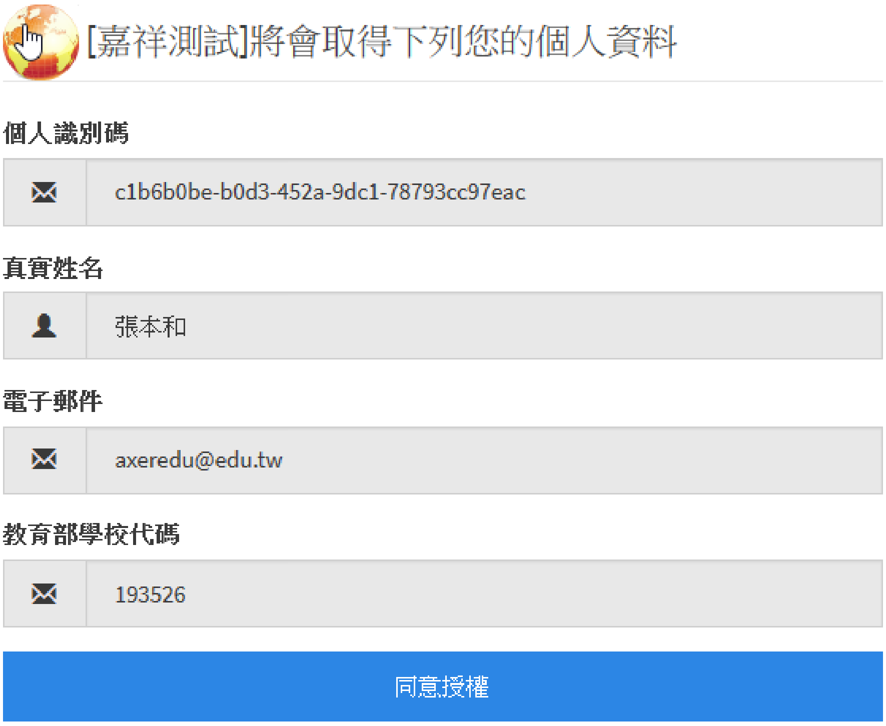


axer(神一般的存在) 目前有100 精幣

username	password	nickname	coins	mappingID
axer	123456	神一般的存在	100	

畫面中的表格資料為資料庫中的欄位與記錄, 為了與教育部帳號服務的帳號綁定, 我們在資料庫中新增 mappingID 的欄位, 底下改用教育部 OIDC 登入,茲以說明如何進行綁定

教育部 OIDC 登入 



[嘉祥測試]將會取得下列您的個人資料

個人識別碼

c1b6b0be-b0d3-452a-9dc1-78793cc97eac

真實姓名

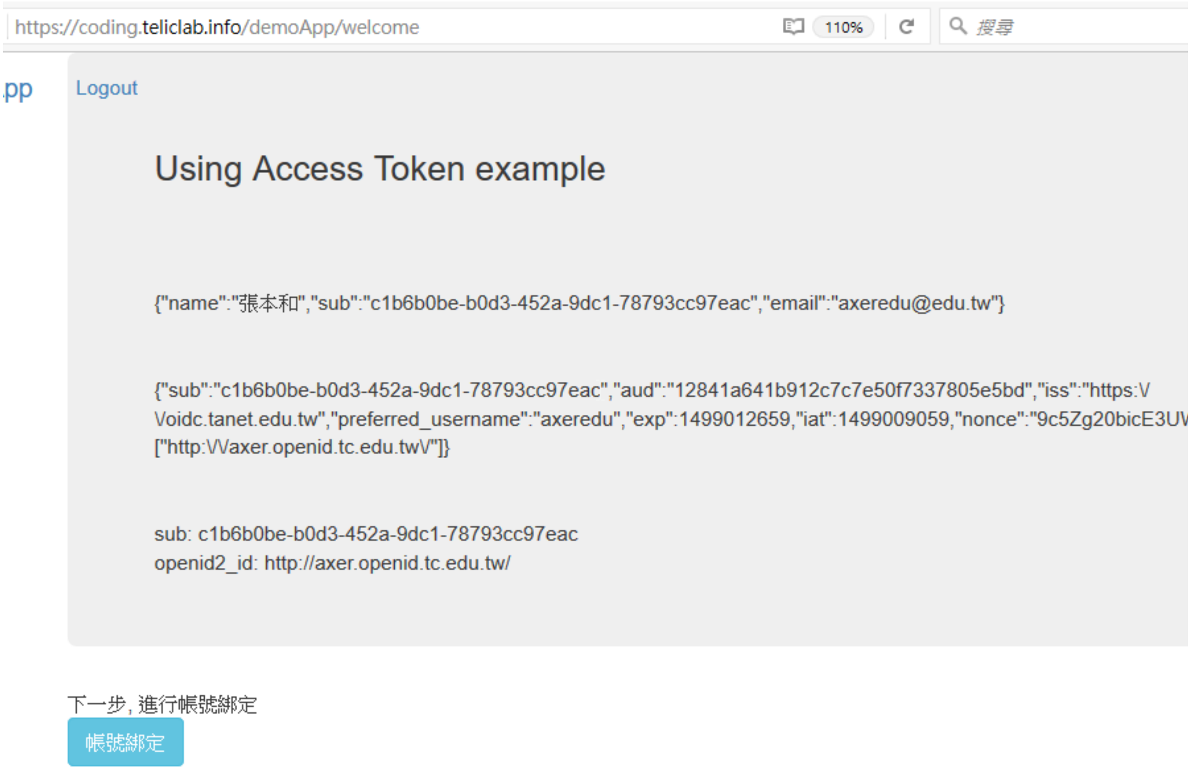
張本和

電子郵件

axeredu@edu.tw

教育部學校代碼

193526



Using Access Token example

{"name":"張本和","sub":"c1b6b0be-b0d3-452a-9dc1-78793cc97eac","email":"axeredu@edu.tw"}

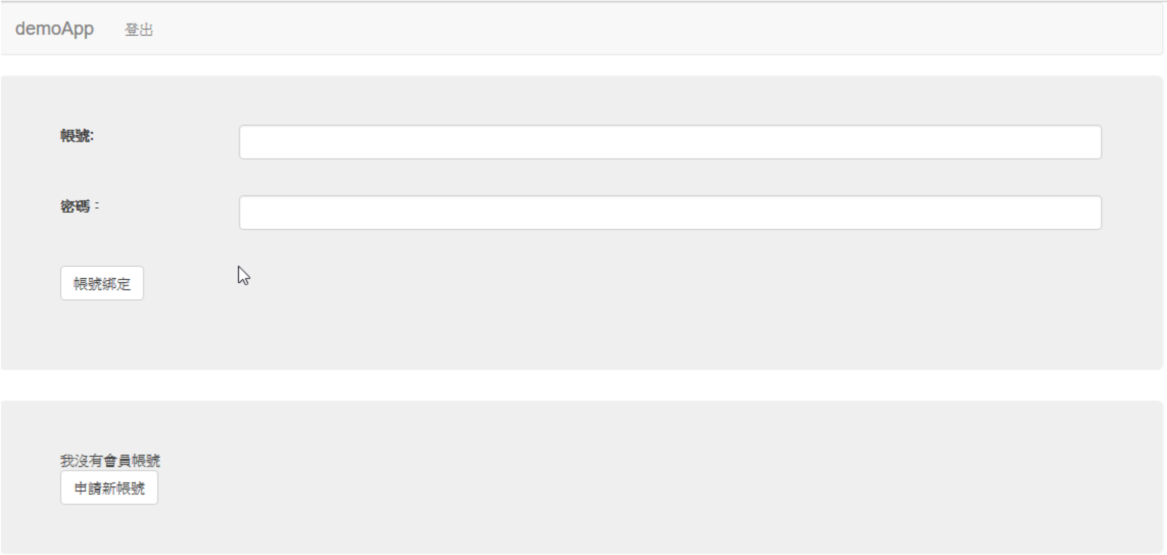
{"sub":"c1b6b0be-b0d3-452a-9dc1-78793cc97eac","aud":"12841a641b912c7c7e50f7337805e5bd","iss":"https:\/
\/oidc.tanet.edu.tw","preferred_username":"axeredu","exp":1499012659,"iat":1499009059,"nonce":"9c5Zg20bicE3UV
["http:\/\/axer.openid.tc.edu.tw\/"]}

sub: c1b6b0be-b0d3-452a-9dc1-78793cc97eac
openid2_id: http://axer.openid.tc.edu.tw/

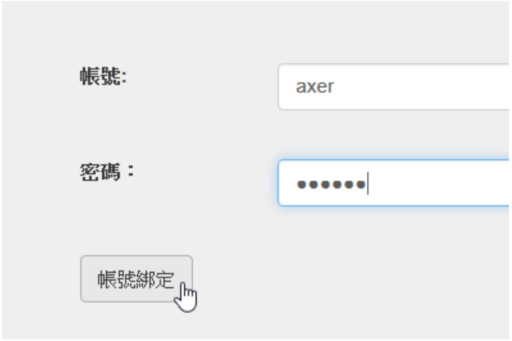
下一步，進行帳號綁定

此為上一章在介接成功並且登入後轉到 dempApp 的 welcome 頁面, 如圖在取得 sub 值及 open2_id 的值後, 即可進行帳號綁定的邏輯判斷, 為了方便說明, 我們把帳號綁定的流程設定為按下按鈕後進行, 在實際上線的主机, 應該在取得值後直接在後端處理再轉向頁面, 可能有三种情況：

1. 未綁定, 輸入會員帳號密碼, 與資料庫比對成功後, 寫入 username 對應的 mappingID 欄位
2. 未綁定, 無會員帳號, 則進行一般的申請新帳號, 並同時寫入 mappingID 欄位
3. 已綁定, sub 值可在 mappingID 中找到, 則取得對應的 username, 並直接轉向 userhome 頁面

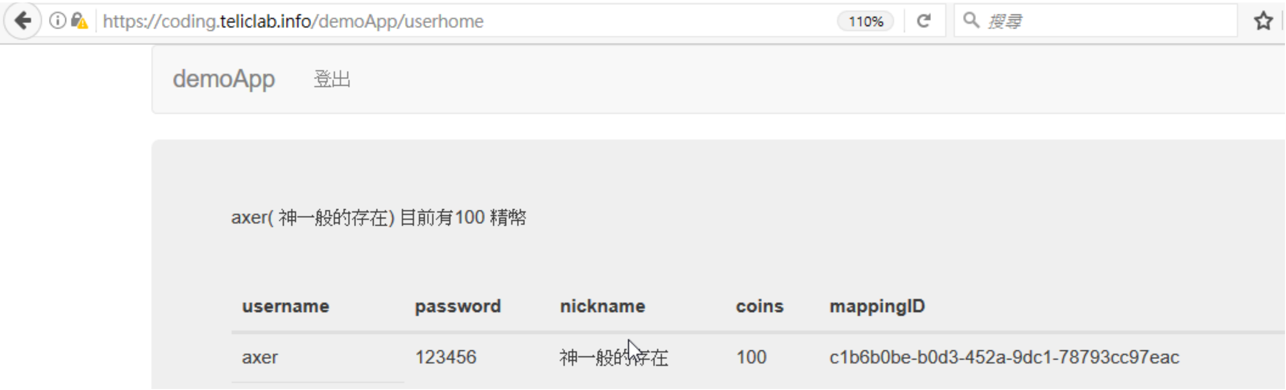


1,2 种情況的的邏輯判斷是相近的, 因此我們以第一种情況做說明


帳號: axer

密碼：••••••

輸入帳號密碼後, 經由資料庫比對成功, 可以由下圖看出在轉向 userhome 頁面後, mappingID 已寫入 sub 值



axer(神一般的存在) 目前有100 精幣

username	password	nickname	coins	mappingID
axer	123456	神一般的存在	100	c1b6b0be-b0d3-452a-9dc1-78793cc97eac

當使用者下次以教育部 OIDC 帳號登入時, 在 welcome 頁面判斷資料庫中的 mappingID 存有此 sub 值, 則依第三种情況帳號已綁定,並找到對應的 username, 並導向該 user 的 userhome, 原先存在資料庫裡的使用者各項資料也能一併帶入