**Appendices**

**Source Code**

**index.jsptharu**

<%@page language="java"

contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1" %>

<!DOCTYPE html>

<html>

<head>

<title>Facebook OAuth Sample Webapp</title>

<script src="//code.jquery.com/jquery-1.11.3.min.js"></script>

<script>

$(document).ready(function () {

$("#FacebookButton").click(makeRequest);

});

function makeRequest() {

// Define properties

var AUTH\_ENDPOINT = "https://www.facebook.com/dialog/oauth";

var RESPONSE\_TYPE = "code";

var CLIENT\_ID = "494738091675788";

var REDIRECT\_URI = "https://localhost:8443/myFacebookoauthapp/callback";

var SCOPE = "public\_profile email";

var STATE = "123";

// Build authorization request endpoint

// According OAuth 2 specification, all the request parameters should be URL encoded

var requestEndpoint = AUTH\_ENDPOINT + "?" +

"response\_type=" + encodeURIComponent(RESPONSE\_TYPE) + "&" +

"client\_id=" + encodeURIComponent(CLIENT\_ID) + "&" +

"redirect\_uri=" + encodeURIComponent(REDIRECT\_URI) + "&" +

"scope=" + encodeURIComponent(SCOPE) + "&" +

"state=" + encodeURIComponent(STATE);

// Send to authorization request endpoint

window.location.href = requestEndpoint;

}

</script>

</head>

<body>

<h1>Signup Account</h1>

<table border="0">

<tr>

<td>

<table border="0">

<tr>

<td>First Name</td>

<td><input type="text" name="first\_name" id="first\_name"/></td>

</tr>

<tr>

<td>Last Name</td>

<td><input type="text" name="last\_name" id="last\_name"/></td>

</tr>

<tr>

<td>Email Address</td>

<td><input type="text" name="email" id="email"/></td>

</tr>

<tr>

<td>ID</td>

<td><input type="text" name="industry" id="id"/></td>

</tr>

<tr>

<td>

<button id="register" type="button">Register</button>

</td>

</tr>

</table>

</td>

<td>

<table border="0">

<tr>

<td><input type="image" src="images/fb-login-button.png" id="FbBtn" width="150"/></td>

</tr>

</table>

</td>

</tr>

</table>

</body>

</html>

**FacebookOAuthAppConstants.java**

package com.ss.oauth.fb;

public class FacebookOAuthAppConstants {

public static final String TOKEN\_ENDPOINT = "https://graph.facebook.com/oauth/access\_token";

public static final String GRANT\_TYPE = "authorization\_code";

public static final String REDIRECT\_URI = "https://localhost:8443/myFacebookoauthapp/callback";

public static final String CLIENT\_ID = "494738091675788";

public static final String CLIENT\_SECRET = "f1e0055dcdef686a0502e6cd2f94d954";

}

**OAuthCallbackListener.java**

package com.ss.oauth.fb;

import java.io.IOException;

import java.net.URLEncoder;

import java.nio.charset.StandardCharsets;

import java.util.HashMap;

import java.util.Map;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.json.simple.JSONObject;

import org.json.simple.parser.ParseException;

import org.json.simple.parser.JSONParser;

import com.ss.oauth.Facebook.util.OAuthTLSUtil;

public class OAuthCallbackListener extends HttpServlet {

private static final long serialVersionUID = 1L;

private final String FacebookUserInfoEndpoint = "https://graph.facebook.com/me?fields~:(first\_name,last\_name,email,id)?format=json";

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Detect the presence of an authorization code

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

if (authorizationCode != null && authorizationCode.length() > 0) {

// Generate POST request

String url = FacebookOAuthAppConstants.TOKEN\_ENDPOINT

+ "?grant\_type=" + URLEncoder.encode(FacebookOAuthAppConstants.GRANT\_TYPE, StandardCharsets.UTF\_8.name())

+ "&code=" + URLEncoder.encode(authorizationCode, StandardCharsets.UTF\_8.name())

+ "&redirect\_uri=" + URLEncoder.encode(FacebookOAuthAppConstants.REDIRECT\_URI, StandardCharsets.UTF\_8.name())

+ "&client\_id=" + URLEncoder.encode(FacebookOAuthAppConstants.CLIENT\_ID, StandardCharsets.UTF\_8.name())

+ "&client\_secret=" + URLEncoder.encode(FacebookOAuthAppConstants.CLIENT\_SECRET, StandardCharsets.UTF\_8.name());

Map<String, String> requestProps = new HashMap<String, String>();

requestProps.put("Content-Type", "application/x-www-form-urlencoded");

String output = OAuthTLSUtil.sendRequest(url, null, requestProps, "POST");

String accessToken = null;

JSONParser parser = new JSONParser();

try {

Object obj = parser.parse(output);

JSONObject jsonb = (JSONObject) obj;

accessToken = jsonb.get("access\_token").toString();

} catch (ParseException e) {

// Handle exception

System.out.println("ERROR: " + e.getMessage());

}

Map<String, String> requestHeaders = new HashMap<String, String>();

requestHeaders.put("Authorization", "Bearer " + accessToken);

String resp = OAuthTLSUtil.sendRequest(FacebookUserInfoEndpoint, requestHeaders, requestProps, "GET");

request.setAttribute("public\_profile", resp);

request.getRequestDispatcher("profile.jsp").forward(request, response);

} else {

// Handle failure

}

}

}

**profile.jsp**

<%@ page import="org.json.simple.parser.JSONParser" %>

<%@ page import="org.json.simple.JSONObject" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Facebook OAuth Sample Webapp</title>

</head>

<body>

<%

// Facebook sends the user profile information in JSON format

String profile\_info = (String) request.getAttribute("public\_profile");

JSONParser parser = new JSONParser();

Object obj = parser.parse(profile\_info);

JSONObject jsonb = (JSONObject) obj;

// Accessing profile attributes from the json object

String email = jsonb.get("email").toString();

String first\_name = jsonb.get("first\_name").toString();

String last\_name = jsonb.get("last\_name").toString();

String id = jsonb.get("id").toString();

%>

<h1>Complete Signup</h1>

<table border="0">

<tr>

<td>First Name</td>

<td><input type="text" name="firstName" id="first\_name" size="100" value="<%=first\_name%>"/></td>

</tr>

<tr>

<td>Last Name</td>

<td><input type="text" name="lastName" id="last\_name" size="100" value="<%=last\_name%>"/></td>

</tr>

<tr>

<td>Email Address</td>

<td><input type=" text" name="emailAddress" id="email" size="100" value="<%=email%>"/></td>

</tr>

<tr>

<td>ID/td>

<td><textarea rows="8" cols="100" name="summary" id="id"><%=id%></textarea></td>

</tr>

<tr>

<td></td>

<td><input type="button" value="Register"/> <td/>

</td>

</tr>

</table>

</body>

</html>

package com.ss.oauth.Facebook.util;

import org.apache.commons.codec.binary.StringUtils;

import javax.net.ssl.HttpsURLConnection;

import javax.net.ssl.SSLContext;

import javax.net.ssl.SSLSocketFactory;

import javax.net.ssl.TrustManagerFactory;

import java.io.\*;

import java.net.MalformedURLException;

import java.net.ProtocolException;

import java.net.URL;

import java.nio.charset.StandardCharsets;

import java.security.KeyManagementException;

import java.security.KeyStore;

import java.security.KeyStoreException;

import java.security.NoSuchAlgorithmException;

import java.security.cert.CertificateException;

import java.util.HashMap;

import java.util.Map;

**OAuthTLSUtil.java**

public class OAuthTLSUtil {

/\*\*

\* Truststore type of the client

\*/

private static String trustStoreType = "JKS";

/\*\*

\* Ttrustmanager type of the client

\*/

private static String trustManagerType = "SunX509";

/\*\*

\* Default transport layer security protocol for client

\*/

private static String protocol = "TLSv1.2";

private static String trustStorePath = "truststore.jks";

private static String trustStorePassword = "wso2carbon";

private static KeyStore trustStore;

private static HttpsURLConnection httpsURLConnection;

private static SSLSocketFactory sslSocketFactory;

private static boolean isInitialized = false;

private static void init() {

InputStream is = null;

try {

trustStore = KeyStore.getInstance(trustStoreType);

File keystoreFile = new File(OAuthTLSUtil.class.getClassLoader().getResource(trustStorePath).getFile());

is = new FileInputStream(keystoreFile);

trustStore.load(is, trustStorePassword.toCharArray());

TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(trustManagerType);

trustManagerFactory.init(trustStore);

// Create and initialize SSLContext for HTTPS communication

SSLContext sslContext = SSLContext.getInstance(protocol);

sslContext.init(null, trustManagerFactory.getTrustManagers(), null);

sslSocketFactory = sslContext.getSocketFactory();

isInitialized = true;

} catch (KeyStoreException e) {

e.printStackTrace();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (CertificateException e) {

e.printStackTrace();

} catch (KeyManagementException e) {

e.printStackTrace();

} finally {

if (is != null) {

try {

is.close();

} catch (IOException e) {

System.out.println("Failed to close file. " + e.getMessage());

}

}

}

}

public static String sendRequest(String link, Map<String, String> requestHeaders, Map<String, String> requestProps, String method) {

if (!isInitialized) {

init();

}

InputStream inputStream = null;

BufferedReader reader = null;

String response = null;

try {

URL url = new URL(link);

httpsURLConnection = (HttpsURLConnection) url.openConnection();

httpsURLConnection.setSSLSocketFactory(sslSocketFactory);

httpsURLConnection.setDoOutput(true);

httpsURLConnection.setDoInput(true);

httpsURLConnection.setRequestMethod(method);

if (requestHeaders != null) {

for (Map.Entry<String, String> entry : requestHeaders.entrySet()) {

httpsURLConnection.setRequestProperty(entry.getKey(), entry.getValue());

}

}

if (requestProps != null) {

for (Map.Entry<String, String> entry : requestProps.entrySet()) {

httpsURLConnection.setRequestProperty(entry.getKey(), entry.getValue());

}

}

inputStream = httpsURLConnection.getInputStream();

reader = new BufferedReader(new InputStreamReader(inputStream, StandardCharsets.UTF\_8));

StringBuilder builder = new StringBuilder();

String line;

while ((line = reader.readLine()) != null) {

builder.append(line);

}

response = builder.toString();

} catch (ProtocolException e) {

e.printStackTrace();

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return response;

}

}