ADAL -> MSAL Migration (Java)

17 March 2023 09:51

//Migration steps - Modify the source/configuration files as provided below

- **<Projectfolder>\pom.xml**
  - Replace adal package with msal
  - update version of oauth2-oidc-sdk package
  - Add the package commons-lang3

ADAL	MSAL

<dependency>
    <groupId>com.microsoft.azure</groupId>
    <artifactId>adal4j</artifactId>
    <version>1.6.0</version>
</dependency>
	<dependency>
	<groupId>com.microsoft.azure</groupId>
	    <artifactId>msal4j</artifactId>
	    <version>1.13.4</version>
	</dependency>

<dependency>
    <groupId>com.nimbusds</groupId>
    <artifactId>oauth2-oidc-sdk</artifactId>
    <version>5.24.1</version>
</dependency>
	<dependency>
	    <groupId>com.nimbusds</groupId>
	    <artifactId>oauth2-oidc-sdk</artifactId>
	    <version>10.5.1</version>
	</dependency>

	<dependency>
	        <groupId>org.apache.commons</groupId>
	                <artifactId>commons-lang3</artifactId>
	        <version>3.4</version>
	  </dependency>

- **<Projectfolder>\src\main\java\com\microsoft\aad\adal4jsample\BasicFilter.Java**
  - Remove following imports from adal4j library
    - import com.microsoft.aad.adal4j.AuthenticationContext;
    - import com.microsoft.aad.adal4j.AuthenticationException;
    - import com.microsoft.aad.adal4j.AuthenticationResult;
    - import com.microsoft.aad.adal4j.ClientCredential;
  - Add the MSAL imports
    - import com.microsoft.aad.msal4j.*;
  - Make following code changes

ADAL	MSAL

private boolean isAuthDataExpired(HttpServletRequest httpRequest) {
  AuthenticationResult authData = AuthHelper.getAuthSessionObject(httpRequest);
  return authData.getExpiresOnDate().before(new Date()) ? true : false;
}
	private boolean isAuthDataExpired(HttpServletRequest httpRequest) {
	  IAuthenticationResult authData = AuthHelper.getAuthSessionObject(httpRequest);
	  return authData.expiresOnDate().before(new Date()) ? true : false;
	}

private String getRedirectUrl(String currentUri, String state, String nonce)
      throws UnsupportedEncodingException {
    String redirectUrl = authority
        + this.tenant
        + "/oauth2/authorize?
response_type=code&scope=directory.read.all&response_mode=form_post&redirect_uri="
        + URLEncoder.encode(currentUri, "UTF-8") + "&client_id="
        + clientId + "&resource=https%3a%2f%2fgraph.microsoft.com"
        + "&state=" + state
        + "&nonce=" + nonce;
 	
    return redirectUrl;
  }
	private String getRedirectUrl(String currentUri, String state, String nonce)
	      throws UnsupportedEncodingException {
	    String redirectUrl = authority
	        + this.tenant
	        + "/oauth2/v2.0/authorize?
	response_type=code&response_mode=form_post&redirect_uri="
	        + URLEncoder.encode(currentUri, "UTF-8") + "&client_id="
	        + clientId + "&scope=https%3a%2f%2fgraph.microsoft.com%2f.default"
	        + "&state=" + state
	        + "&nonce=" + nonce;
	
	    return redirectUrl;
	  }

private void updateAuthDataUsingRefreshToken(HttpServletRequest httpRequest) throws Throwable {
    AuthenticationResult authData =
 	
getAccessTokenFromRefreshToken(AuthHelper.getAuthSessionObject(httpRequest).getRefreshToken());
    setSessionPrincipal(httpRequest, authData);
  }
	private void updateAuthDataUsingRefreshToken(HttpServletRequest httpRequest) throws Throwable {
	        IAuthenticationResult authData =
	
	getAccessTokenFromRefreshToken(httpRequest);
	    setSessionPrincipal(httpRequest, authData);
	  }

public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {
    if (request instanceof HttpServletRequest) {
      HttpServletRequest httpRequest = (HttpServletRequest) request;
      HttpServletResponse httpResponse = (HttpServletResponse) response;
      try {
        String currentUri = httpRequest.getRequestURL().toString();
        String queryStr = httpRequest.getQueryString();
        String fullUrl = currentUri + (queryStr != null ? "?" + queryStr : "");
 	
        // check if user has a AuthData in the session
        if (!AuthHelper.isAuthenticated(httpRequest)) {
          if (AuthHelper.containsAuthenticationData(httpRequest)) {
            processAuthenticationData(httpRequest, currentUri, fullUrl);
          } else {
            // not authenticated
            sendAuthRedirect(httpRequest, httpResponse);
            return;
          }
        }
        if (isAuthDataExpired(httpRequest)) {
          updateAuthDataUsingRefreshToken(httpRequest);
        }
      } catch (AuthenticationException authException) {
        // something went wrong (like expiration or revocation of token)
        // we should invalidate AuthData stored in session and redirect to Authorization server
        removePrincipalFromSession(httpRequest);
        sendAuthRedirect(httpRequest, httpResponse);
        return;
      } catch (Throwable exc) {
        httpResponse.setStatus(500);
        request.setAttribute("error", exc.getMessage());
        request.getRequestDispatcher("/error.jsp").forward(request, response);
      }
    }
    chain.doFilter(request, response);
	public void doFilter(ServletRequest request, ServletResponse response,
	            FilterChain chain) throws IOException, ServletException {
	    if (request instanceof HttpServletRequest) {
	      HttpServletRequest httpRequest = (HttpServletRequest) request;
	      HttpServletResponse httpResponse = (HttpServletResponse) response;
	      try {
	        String currentUri = httpRequest.getRequestURL().toString();
	        String queryStr = httpRequest.getQueryString();
	        String fullUrl = currentUri + (queryStr != null ? "?" + queryStr : "");
	
	        // check if user has a AuthData in the session
	        if (!AuthHelper.isAuthenticated(httpRequest)) {
	          if (AuthHelper.containsAuthenticationData(httpRequest)) {
	            processAuthenticationData(httpRequest, currentUri, fullUrl);
	          } else {
	            // not authenticated
	            sendAuthRedirect(httpRequest, httpResponse);
	            return;
	          }
	        }
	        if (isAuthDataExpired(httpRequest)) {
	          updateAuthDataUsingRefreshToken(httpRequest);
	        }
	      } catch (MsalInteractionRequiredException authException) {
	        // something went wrong (like expiration or revocation of token)
	        // we should invalidate AuthData stored in session and redirect to Authorization server
	        removePrincipalFromSession(httpRequest);
	        sendAuthRedirect(httpRequest, httpResponse);
	        return;
	      } catch (Throwable exc) {
	        httpResponse.setStatus(500);
	        request.setAttribute("error", exc.getMessage());
	        request.getRequestDispatcher("/error.jsp").forward(request, response);
	      }
	    }
	    chain.doFilter(request, response);

}	}

private void processAuthenticationData(HttpServletRequest httpRequest, String currentUri, String fullUrl)
throws Throwable {
HashMap<String, String> params = new HashMap<>();
for (String key : httpRequest.getParameterMap().keySet()) {
params.put(key, httpRequest.getParameterMap().get(key)[0]);
}
 	
// validate that state in response equals to state in request
StateData stateData = validateState(httpRequest.getSession(), params.get(STATE));
 	
AuthenticationResponse authResponse = AuthenticationResponseParser.parse(new URI(fullUrl), params);
if (AuthHelper.isAuthenticationSuccessful(authResponse)) {
AuthenticationSuccessResponse oidcResponse = (AuthenticationSuccessResponse) authResponse;
// validate that OIDC Auth Response matches Code Flow (contains only requested artifacts)
validateAuthRespMatchesCodeFlow(oidcResponse);
 	
AuthenticationResult authData =
getAccessToken(oidcResponse.getAuthorizationCode(), currentUri);
// validate nonce to prevent reply attacks (code maybe substituted to one with broader access)
validateNonce(stateData, getClaimValueFromIdToken(authData.getIdToken(), "nonce"));
 	
setSessionPrincipal(httpRequest, authData);
} else {
AuthenticationErrorResponse oidcResponse = (AuthenticationErrorResponse) authResponse;
throw new Exception(String.format("Request for auth code failed: %s - %s",
oidcResponse.getErrorObject().getCode(),
oidcResponse.getErrorObject().getDescription()));
}
}
	private void processAuthenticationData(HttpServletRequest httpRequest, String currentUri, String fullUrl)
	throws Throwable {
	Map<String, List<String>> params = new HashMap<>();
	for (String key : httpRequest.getParameterMap().keySet()) {
	List<String> lst = new ArrayList<String>();
	lst.add(httpRequest.getParameterMap().get(key)[0]);
	params.put(key,lst);
	}
	// validate that state in response equals to state in request
	StateData stateData = validateState(httpRequest.getSession(), params.get(STATE).get(0));
	
	AuthenticationResponse authResponse = AuthenticationResponseParser.parse(new URI(fullUrl), params);
	if (AuthHelper.isAuthenticationSuccessful(authResponse)) {
	AuthenticationSuccessResponse oidcResponse = (AuthenticationSuccessResponse) authResponse;
	// validate that OIDC Auth Response matches Code Flow (contains only requested artifacts)
	validateAuthRespMatchesCodeFlow(oidcResponse);
	
	IAuthenticationResult authData =
	getAccessToken(oidcResponse.getAuthorizationCode(), currentUri);
	// validate nonce to prevent reply attacks (code maybe substituted to one with broader access)
	validateNonce(stateData, getClaimValueFromIdToken(authData.idToken(), "nonce"));
	
	setSessionPrincipal(httpRequest, authData);
	} else {
	AuthenticationErrorResponse oidcResponse = (AuthenticationErrorResponse) authResponse;
	throw new Exception(String.format("Request for auth code failed: %s - %s",
	oidcResponse.getErrorObject().getCode(),
	oidcResponse.getErrorObject().getDescription()));
	}
	}

private AuthenticationResult getAccessTokenFromRefreshToken(
String refreshToken) throws Throwable {
AuthenticationContext context;
AuthenticationResult result = null;
ExecutorService service = null;
try {
 	
service = Executors.newFixedThreadPool(1);
 	
context = new AuthenticationContext(authority + tenant + "/", true,
service);
 	
Future<AuthenticationResult> future = context
.acquireTokenByRefreshToken(refreshToken, new ClientCredential(clientId, clientSecret), null, null);
 	
result = future.get();
 	
} catch (ExecutionException e) {
throw e.getCause();
} finally {
service.shutdown();
}
 	
if (result == null) {
throw new ServiceUnavailableException("authentication result was null");
}
 	
return result;
}
	private IAuthenticationResult getAccessTokenFromRefreshToken(
	HttpServletRequest httpRequest) throws Throwable {
	
	IAuthenticationResult result = AuthHelper.getAuthSessionObject(httpRequest);
	IClientCredential credential = ClientCredentialFactory.createFromSecret(clientSecret);
	IAuthenticationResult updatedResult = null;
	ExecutorService service = null;
	try {
	service = Executors.newFixedThreadPool(1);
	
	IConfidentialClientApplication app = ConfidentialClientApplication.builder(clientId,
	credential).
	authority(authority).
	build();
	SilentParameters parameters = SilentParameters.builder(
	Collections.singleton("User.Read"),
	result.account()).build();
	
	CompletableFuture<IAuthenticationResult> future = app.acquireTokenSilently(parameters);
	
	updatedResult = future.get();
	
	} catch (ExecutionException e) {
	throw e.getCause();
	} finally {
	service.shutdown();
	}
	
	if (updatedResult == null) {
	throw new ServiceUnavailableException("authentication result was null");
	}
	
	return updatedResult;
	}

//Many changes
private AuthenticationResult getAccessToken(
AuthorizationCode authorizationCode, String currentUri)
throws Throwable {
String authCode = authorizationCode.getValue();
ClientCredential credential = new ClientCredential(clientId,
clientSecret);
AuthenticationContext context;
AuthenticationResult result = null;
ExecutorService service = null;
try {
service = Executors.newFixedThreadPool(1);
context = new AuthenticationContext(authority + tenant + "/", true,
service);
Future<AuthenticationResult> future = context
.acquireTokenByAuthorizationCode(authCode, new URI(
currentUri), credential, null);
 	
result = future.get();
} catch (ExecutionException e) {
throw e.getCause();
} finally {
service.shutdown();
}
 	
if (result == null) {
throw new ServiceUnavailableException("authentication result was null");
}
return result;
}
	private IAuthenticationResult getAccessToken(
	AuthorizationCode authorizationCode, String currentUri)
	throws Throwable {
	String authCode = authorizationCode.getValue();
	IClientCredential credential = ClientCredentialFactory.createFromSecret(clientSecret);
	IConfidentialClientApplication context = null;
	IAuthenticationResult result = null;
	ExecutorService service = null;
	try {
	service = Executors.newFixedThreadPool(1);
	context = ConfidentialClientApplication.builder(clientId,
	credential).
	authority(authority
	+ this.tenant + "/").
	build();
	AuthorizationCodeParameters parameters = AuthorizationCodeParameters.builder(
	authCode,
	new URI(currentUri)).
	build();
	Future<IAuthenticationResult> future = context.acquireToken(parameters);
	
	result = future.get();
	
	} catch (ExecutionException e) {
	throw e.getCause();
	} finally {
	service.shutdown();
	}
	
	if (result == null) {
	throw new ServiceUnavailableException("authentication result was null");
	}
	
	return result;

	}

private void setSessionPrincipal(HttpServletRequest httpRequest,
AuthenticationResult result) {
httpRequest.getSession().setAttribute(AuthHelper.PRINCIPAL_SESSION_NAME, result);
}
	private void setSessionPrincipal(HttpServletRequest httpRequest,
	IAuthenticationResult result) {
	httpRequest.getSession().setAttribute(AuthHelper.PRINCIPAL_SESSION_NAME, result);
	}

- **<Projectfolder>\src\main\java\com\microsoft\aad\adal4jsample\AuthHelper.java**
  - Remove below import from adal4j lib
    - import com.microsoft.aad.adal4j.AuthenticationResult;
  - Remove the below import from nimbusds lib
    - import com.nimbusds.openid.connect.sdk.AuthenticationResponseParser;
  - Add following import from msal4j lib
    - import com.microsoft.aad.msal4j.*;
  - Make following code changes

ADAL	MSAL

public static AuthenticationResult getAuthSessionObject(
HttpServletRequest request) {
return (AuthenticationResult) request.getSession().getAttribute(
PRINCIPAL_SESSION_NAME);
}
	public static IAuthenticationResult getAuthSessionObject(
	HttpServletRequest request) {
	return (IAuthenticationResult) request.getSession().getAttribute(
	PRINCIPAL_SESSION_NAME);
	}

- **<Projectfolder>\src\main\java\com\microsoft\aad\adal4jsample\AadController.java**
  - Remove following import from adal4j lib
    - import com.microsoft.aad.adal4j.AuthenticationResult;
  - Add following import from msal4j lib
    - import com.microsoft.aad.msal4j.*;
  - Add following additional imports (to support extracting claims from idtoken)
    - import java.io.IOException;
    - import com.nimbusds.jwt.SignedJWT;
    - import com.nimbusds.jwt.JWTClaimsSet;
    - import java.util.*;
  - Make following code changes
  - M

ADAL	MSAL

@RequestMapping(method = { RequestMethod.GET, RequestMethod.POST })
public String getDirectoryObjects(ModelMap model, HttpServletRequest httpRequest) {
HttpSession session = httpRequest.getSession();
AuthenticationResult result = (AuthenticationResult)
session.getAttribute(AuthHelper.PRINCIPAL_SESSION_NAME);
if (result == null) {
model.addAttribute("error", new Exception("AuthenticationResult not found in session."));
return "/error";
} else {
String data;
try {
String tenant = session.getServletContext().getInitParameter("tenant");
data = getUsernamesFromGraph(result.getAccessToken(), tenant);
model.addAttribute("tenant", tenant);
model.addAttribute("users", data);
model.addAttribute("userInfo", result.getUserInfo());
} catch (Exception e) {
model.addAttribute("error", e);
return "/error";
}
}
return "/secure/aad";
}
	@RequestMapping(method = { RequestMethod.GET, RequestMethod.POST })
	public String getDirectoryObjects(ModelMap model, HttpServletRequest httpRequest) {
	HttpSession session = httpRequest.getSession();
	IAuthenticationResult result = (IAuthenticationResult)
	session.getAttribute(AuthHelper.PRINCIPAL_SESSION_NAME);
	if (result == null) {
	model.addAttribute("error", new Exception("AuthenticationResult not found in session."));
	return "/error";
	} else {
	String data;
	try {
	String tenant = session.getServletContext().getInitParameter("tenant");
	data = getUsernamesFromGraph(result.accessToken(), tenant);
	model.addAttribute("tenant", tenant);
	model.addAttribute("users", data);
	model.addAttribute("userInfo", getUserInfoFromGraph(result));
	} catch (Exception e) {
	model.addAttribute("error", e);
	return "/error";
	}
	}
	return "/secure/aad";
	}

	private JSONObject getUserInfoFromGraph(IAuthenticationResult result) throws Exception {
	// Microsoft Graph user endpoint
	String accessToken = result.accessToken();
	URL url = new URL("https://graph.microsoft.com/v1.0/me");
	HttpURLConnection conn = (HttpURLConnection) url.openConnection();
	
	// Set the appropriate header fields in the request header.
	conn.setRequestProperty("Authorization", "Bearer " + accessToken);
	conn.setRequestProperty("Accept", "application/json");
	
	String response = HttpClientHelper.getResponseStringFromConn(conn,true);
	
	int responseCode = conn.getResponseCode();
	if(responseCode != HttpURLConnection.HTTP_OK) {
	throw new IOException(response);
	}
	
	JSONObject responseObject = HttpClientHelper.processGoodRespStr(responseCode, response);
	JSONObject userObject = JSONHelper.fetchDirectoryObjectJSONObject(responseObject);
	
	SignedJWT signedJWT = SignedJWT.parse(result.idToken());
	JWTClaimsSet claimsSet= signedJWT.getJWTClaimsSet();
	Map<String,Object> myClain =claimsSet.getClaims();
	
	String stridp = "";
	
	if(myClain.get("idp")!=null)
	{
	stridp = myClain.get("idp").toString();
	}
	
	userObject.put("identityProvider",stridp);
	
	return userObject;
	}

- **<Projectfolder>\src\main\webapp\secure\aad.jsp**
  - Make following changes for jsp tags

ADAL	MSAL
<!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=UTF-8"> <title>AAD Secure Page</title> </head> <body> <h3>List of users registered in the tenant - ${tenant}</h3> <p>${users}</p> <h3>Current user</h3> <table> <tr> <td>uniqueId:</td> <td>${userInfo.uniqueId}</td> </tr> <tr> <td>displayableId:</td> <td>${userInfo.displayableId}</td> </tr> <tr> <td>givenName:</td> <td>${userInfo.givenName}</td> </tr> <tr> <td>familyName:</td> <td>${userInfo.familyName}</td> </tr> <tr> <td>identityProvider:</td> <td>${userInfo.identityProvider}</td> </tr> </table> <form action="/adal4jsample"> <input type="submit" value="Home Page"> </form> </body> </html>	