Custom Authenticator and Login Module in native Windows 8 applications

fork and edit tutorial (https://github.ibm.com/MFPSamples/DevCenter/tree/master/tutorials/en/foundation/7.0/authentication-security/custom-authenticator-login-module/custom-authenticator-login-module-native-windows-8-applications.html) | report issue (https://github.ibm.com/MFPSamples/DevCenter/issues/new)

Overview

This tutorial illustrates the native Windows 8 client-side authentication components for custom authentication. Make sure you read Custom Authenticator and Login Module (../) first.

Creating the client-side authentication components

Create a native Windows 8 application and add the MobileFirst native APIs following the documentation.

CustomChallengeHandler

Create a CustomChallengeHandler class as a subclass of ChallengeHandler. CustomChallengeHandler should implement

- isCustomResponse
- handleChallenge

isCustomResponse checks every custom response received from MobileFirst Server to see if this is the challenge we are expecting.

```
public override bool isCustomResponse(WLResponse response)
{
   if (!(response.getResponseJSON()["authStatus"] == null) && response.getResponseJSON()["authStatus"].ToString
().CompareTo("required") == 0)
   {
      return true;
   }
   else
   {
      return false;
   }
}
```

handleChallenge method, is called after the <code>isCustomResponse</code> method returned true.

Within this method we present our login form. Different approaches may be adopted to present the login form.

```
public override void handleChallenge(JObject response)
{
   CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPriority.Normal
,
   async () =>
   {
      MainPage._this.LoginGrid.Visibility = Visibility.Visible;
   });
}
```

From the login form, credentials are passed to the CustomChallengeHandler class. The submitLoginForm() method is used to send our input data to the authenticator.

```
public void sendResponse(String username, String password)
{
    Dictionary<String, String> parms = new Dictionary<String, String>();
    parms.Add("username", username);
    parms.Add("password", password);
    submitLoginForm("/my_custom_auth_request_url", parms, null, 0, "post")
;
}
```

MainPage

Within the MainPage class connect to MobileFirst server, register your challengeHandler and invoke the protected adapter procedure.

The procedure invocation will trigger MobileFirst server to send a challenge that will trigger our challengeHandler.

```
WLClient wlClient = WLClient.getInstance();
CustomChallengeHandler ch = new CustomChallengeHandler();
wlClient.registerChallengeHandler((BaseChallengeHandler<JObject>)ch);
MyResponseListener mylistener = new MyResponseListener(this);
wlClient.connect(mylistener);
```

Since the native API not protected by a defined security test, there is no login form presented during server connection. Invoke the protected adapter procedure and the login form is presented by the challengeHandler.

```
WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");
Object[] parameters = { 0 };
invocationData.setParameters(parameters);
MyInvokeListener listener = new MyInvokeListener(this);
WLClient.getInstance().invokeProcedure(invocationData, listener, new WLRequestOptions());
```

Sample application

Click to download

(http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/NativeCustomLoginModuleProject.zip) the Studio project.

Click to download

(http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v700/Win8NativeCustomLoginModuleProject.zip) the Native project.

