

Form-based authentication in native Windows 8 applications

Overview

This tutorial illustrates the native Windows 8 Universal client-side authentication components for form-based authentication.

Prerequisite: Make sure that you read Form-based authentication (../) first.

This tutorial covers the following topics:

- Creating the client-side authentication components
- Sample application

Creating the client-side authentication components

Create a native Windows 8 Universal application and add the MobileFirst native APIs as explained in the documentation.

FormChallengeHandler

Create a `FormChallengeHandler` class as a subclass of `ChallengeHandler`.

Your `FormChallengeHandler` class must implement the `isCustomResponse` and `handleChallenge` methods.

The `isCustomResponse` method checks every custom response received from MobileFirst Server to verify whether this is the expected challenge.

```
1  public override bool isCustomResponse(WLResponse response)
2  {
3      if (response == null || response.GetResponseText() == null || !response.GetResponseText().Contain
4      {
5          return false;
6      }
7      else
8      {
9          return true;
10     }
11 }
```

The `handleChallenge` method is called after the `isCustomResponse` method returns `true`. Within this method, present your login form. Different approaches are available.

```

1 public override void handleChallenge(JObject response)
2 {
3     CoreApplication.MainView.CoreWindow.Dispatcher.RunAsync(CoreDispatcherPriority.Normal,
4     async () =>
5     {
6         MainPage._this.LoginGrid.Visibility = Visibility.Visible;
7     });
8 }

```

From the login form , credentials are passed to the FormChallengeHandler class. Use the submitLoginForm() method to send input data to the authenticator.

```

1 public void sendResponse(String username, String password)
2 {
3     Dictionary<String, String> parms = new Dictionary<String, String>();
4     parms.Add("j_username", username);
5     parms.Add("j_password", password);
6     submitLoginForm("j_security_check", parms, null, 0, "post");
7 }

```

MainPage

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

The procedure invocation triggers MobileFirst Server to send a challenge that will trigger our challenge handler.

```

1 WLClient wClient = WLClient.getInstance();
2 FormChallengeHandler ch = new FormChallengeHandler();
3 wClient.registerChallengeHandler((BaseChallengeHandler<JObject>)ch);
4 MyResponseListener mylistener = new MyResponseListener(this);
5 wClient.connect(mylistener);

```

Because the native API not protected by a defined security test, no login form is presented during server connection.

Invoke the protected adapter procedure. The login form is presented by the challengeHandler.

```

1 WLResourceRequest adapter = new WLResourceRequest("/adapters/AuthAdapter/getSecretData", "GET")
2 MyInvokeListener listener = new MyInvokeListener(this);
3 adapter.send(listener);

```

Sample application

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/FormBasedAuth>) the MobileFirst project.

Click to download (<https://github.com/MobileFirst-Platform-Developer-Center/FormBasedAuthWin8>) the Native project.

- The FormBasedAuth project contains a MobileFirst native API that you can deploy to your MobileFirst server.
- The FormBasedAuthWin8 project contains a native Windows 8 Universal application that uses a MobileFirst native API library.
- Make sure to update the `wlclient.properties` file in the native project with the relevant server settings.

