

Form-based authentication in native Windows Phone 8 applications

Overview

This tutorial illustrates the native Windows Phone 8 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 Phone 8 application and add the MobileFirst native APIs as explained in the documentation.

MyChallengeHandler

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

Your `FormChallengeHandler` class must implement the `isCustomResponse` method.

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 ||
4         response.GetResponseText() == null ||
5         !response.GetResponseText().Contains("j_security_check"))
6     {
7         return false;
8     }
9     else
10    {
11        return true;
12    }
13 }
```

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

```

1  public override void handleChallenge(JObject response)
2  {
3      Deployment.Current.Dispatcher.BeginInvoke() =>
4      {
5          MainPage._this.NavigationService.Navigate(new Uri("/LoginPage.xaml", UriKind.Relative));
6      });
7  }

```

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 submit(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, 10000, "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 the challenge handler.

```

1  WLClient client;
2  client = WLClient.getInstance();
3  challengeHandler = new WindowsChallengeHandler();
4  client.registerChallengeHandler((BaseChallengeHandler<JObject>)challengeHandler);
5  client.connect(new MyConnectResponseListener(this));

```

Because the native API is 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  WLProcedureInvocationData invocationData = new WLProcedureInvocationData("AuthAdapter", "getSec
2  invocationData.setParameters(new Object[] { });
3  WLRequestOptions options = new WLRequestOptions();
4  WLClient.getInstance().invokeProcedure(invocationData, new MyResponseListener(this), options);

```

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/FormBasedAuthWP8>) the Native project.

- The FormBasedAuth project contains a MobileFirst native API that you can deploy to your MobileFirst server.
- The FormBasedAuthWP8 project contains a native WP8 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.

