

Form-based authentication in native Android applications

Overview

This tutorial illustrates the native Android client-side authentication components for form-based authentication. Make sure you read [Form-based authentication \(../\)](#) first.

Creating the client-side authentication components

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

Add an Activity, `LoginFormBasedAuth`, that will handle and present the login form.

Remember to add this Activity to the `AndroidManifest.xml` file as well.

MyChallengeHandler

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

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

```

1  public boolean isCustomResponse(WLResponse response) {
2      if (response == null || response.getResponseText() == null ||
3          response.getResponseText().indexOf("_security_check") == -1) {
4          return false;
5      }
6      return true;
7  }
```

`handleChallenge` is called after the `isCustomResponse` method returned true.

Here we use this method to present our login form.

```

1  public void handleChallenge(WLResponse response){
2      if (!isCustomResponse(response)) {
3          submitSuccess(response);
4      } else {
5          cachedResponse = response;
6          Intent login = new Intent(parentActivity, LoginFormBasedAuth.class);
7          parentActivity.startActivityForResult(login, 1);
8      }
9  }
```

`submitLogin` is called by the login form. If the user asked to abort this action we use `submitFailure()` method, otherwise we use `submitLoginForm()` method to send our input data to the authenticator.

```

1  public void submitLogin(int resultCode, String userName, String password, boolean back){
2      if (resultCode != Activity.RESULT_OK || back) {
3          submitFailure(cachedResponse);
4      } else {
5          HashMap<String, String> params = new HashMap<String, String>();
6          params.put("_username", userName);
7          params.put("_password", password);
8          submitLoginForm("/_security_check", params, null, 0, "post");
9      }
10 }
```

Main Activity

In the Main Activity 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.

```
1 final WLCClient client = WLCClient.createInstance(this);
2 client.connect(new MyConnectionListener());
3 challengeHandler = new AndroidChallengeHandler(this, realm);
4 client.registerChallengeHandler(challengeHandler);
5 invokeBtn = (Button) findViewById(R.id.invoke);
6 invokeBtn.setOnClickListener(new View.OnClickListener() {
7     @Override
8     public void onClick(View v) {
9         //setMainText("Invoking...");
10        WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter", "getSecretData");
11        WLRequestOptions options = new WLRequestOptions();
12        options.setTimeout(30000);
13        client.invokeProcedure(invocationData, new MyResponseListener(), options);
14    }
15 });
```

Sample application

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/NativeFormBasedAuthProject.zip>) the Studio project.

Click to download

(<http://public.dhe.ibm.com/software/products/en/MobileFirstPlatform/docs/v630/AndroidNativeFormBasedAuthProject.zip>) the Native project.

