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.

Copy

```
public boolean isCustomResponse(WLResponse response) {
 if (response == null || response.getResponseText() == null ||
   response.getResponseText().indexOf("j_security_check") == -1)
{
  return false;
 }
 return true;
}
```

`handleChallenge` is called after the `isCustomResponse` method returned true.
Here we use this method to present our login form.

Copy

```
public void handleChallenge(WLResponse response){
 if (!isCustomResponse(response)) {
  submitSuccess(response);
 } else {
  cachedResponse = response;
  Intent login = new Intent(parentActivity,
LoginFormBasedAuth.class);
  parentActivity.startActivityForResult(login, 1);
 }
}
```

`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.

Copy

```
public void submitLogin(int resultCode, String userName, String password, boolean back){
 if (resultCode != Activity.RESULT_OK || back) {
  submitFailure(cachedResponse);
 } else {
  HashMap<String, String> params = new HashMap<String, String>();
  params.put("j_username", userName);
  params.put("j_password", password);
  submitLoginForm("/j_security_check", params, null, 0, "post");
 }
}
```

Main Activity

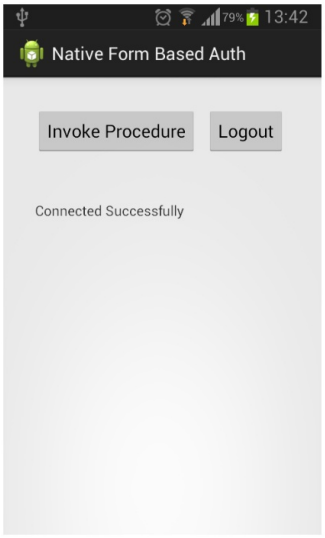
In the Main Activity class connect to MobileFirst server, register your `challengeHandler` and invoke the protected adapter procedure.
The procedure invocation will trigger MobileFirst server to send a challenge that will trigger our `challengeHandler`.

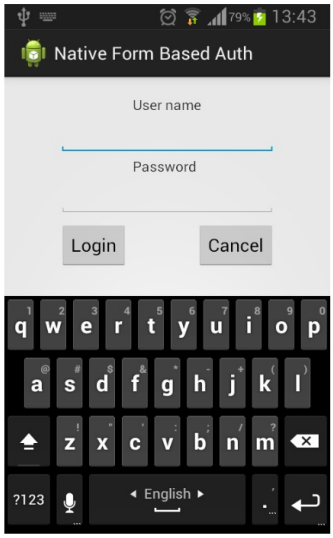
Copy

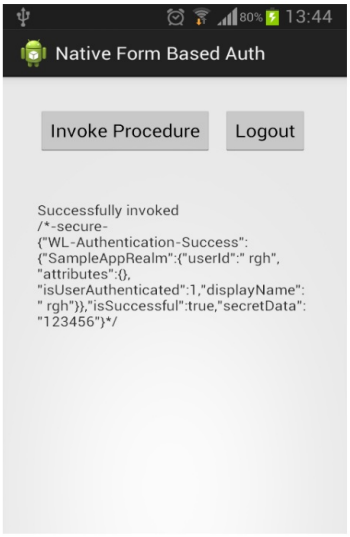
```
final WLClient client = WLClient.createInstance(this);
client.connect(new MyConnectionListener());
challengeHandler = new AndroidChallengeHandler(this, realm);
client.registerChallengeHandler(challengeHandler);
invokeBtn = (Button) findViewById(R.id.invoke);
invokeBtn.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View v) {
  //setMainText("Invoking...");
  WLProcedureInvocationData invocationData = new WLProcedureInvocationData("DummyAdapter",
"getSecretData");
  WLRequestOptions options = new WLRequestOptions();
  options.setTimeout(30000);
  client.invokeProcedure(invocationData, new MyResponseListener(), options);
 }
});
```

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.


13:42
Native Form Based Auth
Invoke Procedure
Logout
Connected Successfully


13:43
Native Form Based Auth
User name
Password
Login
Cancel
English


13:44
Native Form Based Auth
Invoke Procedure
Logout
Successfully invoked
/*-secure-
{"WL-Authentication-Success":
{"SampleAppRealm":{"userId":" rgh",
"attributes":{},
"isUserAuthenticated":1,"displayName":
" rgh"}},"isSuccessful":true,"secretData":
"123456"}*/