

Adapter-based authentication in native Android applications

fork and edit tutorial (<https://github.com/MobileFirst-Platform-Developer-Center/DevCenter/#fork-destination-box>) | report issue (<https://github.com/MobileFirst-Platform-Developer-Center/DevCenter/issues/new>)

This is a continuation of the Adapter-based authentication (../) tutorial.

Creating the client-side authentication components

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

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

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

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

`isCustomResponse` checks every custom response received from MobileFirst Server to see if that's the challenge we are expecting. In the example adapter code a `authRequired` variable is sent for this purpose.

```
public boolean isCustomResponse(WLResponse response) {
    try {
        if(response!= null&&
            response.getResponseJSON()!=null &&
            response.getResponseJSON().isNull("authRequired") != true &&
            response.getResponseJSON().getBoolean("authRequired") == true)
        {
            return true;
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return false;
}
```

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

```
public void handleChallenge(WLResponse response){
    cachedResponse = response;
    Intent login = new Intent(parentActivity, LoginAdapterBasedAuth.class);
    parentActivity.startActivityForResult(login, 1);
}
```

In `submitLogin`, if the user asked to abort this action we use the `submitFailure()` method, otherwise we invoke our adapter authentication procedure using the `submitAdapterAuthentication()` method.

```

public void submitLogin(int resultCode, String userName, String password, boolean back){
    if (resultCode != Activity.RESULT_OK || back) {
        submitFailure(cachedResponse);
    } else {
        Object[] parameters = new Object[]{userName, password};
        WLProcedureInvocationData invocationData = new WLProcedureInvocationData("NativeAdapterBasedAdapter", "submitAuthentication");
        invocationData.setParameters(parameters);
        WLRequestOptions options = new WLRequestOptions();
        options.setTimeout(30000);
        submitAdapterAuthentication(invocationData, options);
    }
}

```

In the Main Activity class, connect to the MobileFirst server, register your `challengeHandler` and invoke the protected adapter procedure.

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

```

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) {
        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/NativeAdapterBasedAuthProject.zip>) the Studio project.

Click to download

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

