Adapter-based authentication in native Android applications

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.

```
1  public boolean isCustomResponse(WLResponse response) {
2   try {
3    if(response!= null&&
4      response.getResponseJSON()!=null &&
5      response.getResponseJSON().isNull("authRequired") != true &&
6      response.getResponseJSON().getBoolean("authRequired") == true){
7     return true;
8    }
9   } catch (JSONException e) {
10   e.printStackTrace();
11  }
12  return false;
13 }
```

`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   cachedResponse = response;
3   Intent login = new Intent(parentActivity, LoginAdapterBasedAuth.class);
4   parentActivity.startActivityForResult(login, 1);
5  }
```

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.

```
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    Object[] parameters = new Object[]{userName, password};
6    WLProcedureInvocationData invocationData = new  WLProcedureInvocationData("NativeAdapterBasedAdapter", "submitAuther
7    invocationData.setParameters(parameters);
8    WLRequestOptions options = new WLRequestOptions();
9    options.setTimeout(30000);
10   submitAdapterAuthentication(invocationData, options);
11  }
12 }
```

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.