Initializing Your Games Client in Android

The `GamesClient` object is used to communicate with the Google Play service. Before making any calls to it, you must first establish an asynchronous connection with the Google Play service.

Typically, you want to:

- Extend the `BaseGameActivity` class (available in the samples).
- Add a **Sign in with Google** button.
- Add a Sign out button.
- When the sign-in button is clicked, initiate the sign in flow.
- When the sign-out button is clicked, initiate the sign out flow.

Download the `BaseGameUtils` project by following the link from the Downloads page.

To import this project and link it to your game in Eclipse:

1. Click **File** > **Import** and select **Existing Android Code into Workspace**. This option is located under the "Android" folder. Click **Next**.
2. Locate the `BaseGameUtils` project folder.
3. Click **Next** and **Finish** to finish the import process.
4. Right-click your game project and select **Properties**.
5. Select **Android** in the list.
6. In the **Library** panel, click the **Add** button.
7. Select `BaseGameUtils`.
8. Click **OK** to finish.

Extend the `BaseGameActivity` class to write your Activity class:

```
public class MainActivity extends BaseGameActivity
    implements View.OnClickListener {
```

The `View.OnClickListener` interface allows us to listen for button click events.

Next, add the sign-in and sign-out buttons to your layout:

```
<!-- sign-in button -->
<com.google.android.gms.common.SignInButton
    android:id="@+id/sign_in_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<!-- sign-out button -->
<Button
    android:id="@+id/sign_out_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sign Out"
    android:visibility="gone" />
```

The exact location of these buttons on your interface is up to you. Notice that the initial visibility of the sign-out button is set to `gone`. It should only be made visible once the user is signed in.

> **Note:** A similar sign-in button is provided by the Google+ Platform for Android. However, for simplicity, we recommend that you use the code example provided here instead.

Next, set up the click callbacks for the sign-in and sign-out buttons on `Activity.onCreate()`:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    findViewById(R.id.sign_in_button).setOnClickListener(this);
    findViewById(R.id.sign_out_button).setOnClickListener(this);
}
```

Implement the `onClick()` callback:

```
@Override
public void onClick(View view) {
    if (view.getId() == R.id.sign_in_button) {
        // start the asynchronous sign in flow
        beginUserInitiatedSignIn();
    }
    else if (view.getId() == R.id.sign_out_button) {
        // sign out.
        signOut();

        // show sign-in button, hide the sign-out button
        findViewById(R.id.sign_in_button).setVisibility(View.VISIBLE);
        findViewById(R.id.sign_out_button).setVisibility(View.GONE);
    }
}
```

The sign in flow is asynchronous. To be notified when the user successfully signed in, override the base class's `onSignInSucceeded()` method:

```
protected void onSignInSucceeded() {
    // show sign-out button, hide the sign-in button
    findViewById(R.id.sign_in_button).setVisibility(View.GONE);
    findViewById(R.id.sign_out_button).setVisibility(View.VISIBLE);

    // (your code here: update UI, enable functionality that depends on sign in,
}
```

To be notified when sign in fails, override `onSignInFailed()`:

```
@Override
protected void onSignInFailed() {
    // Sign in has failed. So show the user the sign-in button.
    findViewById(R.id.sign_in_button).setVisibility(View.VISIBLE);
    findViewById(R.id.sign_out_button).setVisibility(View.GONE);
}
```

> **Note:** Not all calls to `onSignInFailed()` indicate errors. That method may also be called to indicate that automatic sign in needs to show a consent dialog to the user. This situation will be handled automatically once `onBeginUserInitiatedSignIn()` is called, which should happen in response to the user clicking the "Sign in" button.

Guard all calls to `GamesClient` methods with a check to verify that the user is signed in. To do that, verify the value of the `mSignedIn` member variable of the base class:

```
if (mSignedIn) {
    // call a GamesClient method
}
else {
    // alternative implementation (or warn user that they must
    // sign in to use this feature)
}
```