Facebook SDK Manual

Este manual tem como objectivo partilhar conhecimento sobre o FacebookSDK em Android. Nenhum destes exemplos deve ser considerada a melhor prática de programação uma vez que o objectivo deste é mesmo demonstrar como fazer e o que usar. Dar ferramentas para que o utilizador possa programa-las de acordo com aquilo que se considera ser bom programador.

Universidade de Coimbra

Departamento Engenharia Informática

Mestrado Engenharia Software

Computação Móvel

16/11/2015

Erbi Silva (2014186442)

Marco Pereira (2009114979)

Celso Mendes (2009109378)

Índice

Facebook SDK - Iniciação

1 – Aceder a https://developers.facebook.com/

2 – Canto superior direito e, para quem não esteja com o "login" efectuado, deverá efectua-lo.

3 – Com o "log in" realizado, ir a "My Apps" e "Add a new App"

4 – Escolher plataforma "Android"

5 – **Os passos seguintes servem para adicionar o Facebook SDK ao projecto**

5.1 – Ir ao "Android Studio", criar novo projecto com o SDK mínimo 15.

5.2 – Selecionar "API 15: Android 4.0.3" ou versão mais elevada e, criar projecto

5.3 – Com o projecto criado, dirigir-se ao ficheiro ".gradle" (Normalmente é o primeiro e diz "Project: |NomeProjeto|")

5.4 - Adicionar antes de "dependencies" a seguinte linha:

```
repositories { mavenCentral() }
```

5.5 – Fazer o "build" do projecto. Agora já se pode importar o com.facebook.FacebookSDK para a aplicação.

5.6 – Abrir o outro ficheiro ".gradle" e adicionar a seguinte linha:

```
compile 'com.facebook.android:facebook-android-sdk:4.6.0'
```

6 – Colocar a "Package Name" e a "Default Activity Class Main". Estes podem ser encontrados no "Android Manifest". No seguinte exemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.blowstuff.primeiroteste" >

        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

Package Name: org.blowstuff.primeiroteste

Default Activity Class Main: org.blowstuff.primeiroteste.MainActivity

7 – Clicar em "Next" e "Use this package name"

Key Hashes

A geração das “key hashes” pode ser um grande problema na criação de projectos Android. Isto, porque de acordo com os detalhes presentes, é dito para se utilizar o openssl.

Nem sempre a geração da chave pelo openssl irá funcionar, isto verifica-se em alguns casos em que a aplicação esteja a correr em dispositivos diferentes. Derivado a isso, o próximo passo é criar uma “key hash” de forma dinâmica com ajuda da seguinte classe:

```
public class MyApplication extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
        printHashKey();
    }

    public void printHashKey(){
        // Add code to print out the key hash
        try {
            PackageInfo info = getPackageManager().getPackageInfo(
                    "PACKAGE_NAME",
                    PackageManager.GET_SIGNATURES);
            for (Signature signature : info.signatures) {
                MessageDigest md = MessageDigest.getInstance("SHA");
                md.update(signature.toByteArray());
                Log.d("KeyHash:", Base64.encodeToString(md.digest(),
Base64.DEFAULT));
            }
        } catch (PackageManager.NameNotFoundException e) {

        } catch (NoSuchAlgorithmException e) {

        }
    }
}
```

A classe MyApplication é responsável por gerar uma chave de acordo com a informação da package por isso é **<u>importante indicar o nome correcto da package</u>**

- PackageInfo info = getPackageManager().getPackageInfo("PACKAGE_NAME", …)

8 – Criar a classe “MyApplication” e, ir ao Android Manifest. Acrescentar **antes** da linha **<activity android:name …** o seguinte:

```
<application
    android:name=".MyApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme" >
```

9 – Correr o projecto e copiar a KeyHash do "log cat”. **<u>Estamos prontos para criar uma aplicação!</u>**

Log In

Este capitulo vai detalhar como realizar o Log in no facebook. Ara isso vão ser explicados que Permissões, botões e alguns detalhes de código.

1. Inicializar o FacebookSDK, na mainActivity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    FacebookSdk.sdkInitialize(getApplicationContext());
    setContentView(R.layout.activity_main);
}
```

Nota: a linha de código deve ser colocada sempre antes do setContentView(…).

2. Ir à pasta layout e modificar o .xml:

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.facebook.login.widget.LoginButton
        android:id="@+id/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="30dp"
        android:layout_marginBottom="30dp"
</LinearLayout>
```

3. Acrescentar o método onClick com o nome do método chamado “LoginClick”. Criar o método:

```
public void LoginClick(View view) {
    LoginManager.getInstance().logInWithReadPermissions(this,
Arrays.asList("public_profile, publish_actions"));
    LoginManager.getInstance().registerCallback(callbackManager, new
FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {

            //publishOnMyFeed();
            accessToken = AccessToken.getCurrentAccessToken();
        }

        @Override
        public void onCancel() {

        }

        @Override
        public void onError(FacebookException error) {

        }
    });
}
```

4. Inicializar o callback manager. Colocar uma variável global e, inicializar no método onCreate:

```
public class MainActivity extends AppCompatActivity {
    CallbackManager callbackManager;

(…)

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    FacebookSdk.sdkInitialize(getApplicationContext());
    setContentView(R.layout.activity_main);
    callbackManager = CallbackManager.Factory.create();
}
```

5. Criar o método onActivityResult e dizer o que o callback manager deverá realizar:

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    callbackManager.onActivityResult(requestCode, resultCode, data);
}
```

6. Correr aplicação e testar o Login. Erro, mas porque?

Permissões

A grande dor de cabeça do FacebookSDK é o facto de que precisamos de permissões para tudo. Em primeiro lugar, precisamos de dar a nós mesmos a permissão para aceder à Internet. Logo, ir ao Android Manifest e colocar:

```
<uses-permission android:name="android.permission.INTERNET" />
```

De seguida, devemos indicar à nossa aplicação que é uma Facebook Activity, logo:

```
<activity
    android:name="com.facebook.FacebookActivity"
 android:configChanges=
"keyboard|keyboardHidden|screenLayout|screenSize|orientation"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Translucent.NoTitleBar" />
```

De seguida, ligar a aplicação do projecto ao que foi criado no FacebookDevelopers. Para isso, devemos nos dirigir ao site indicado no ponto 1 novamente e, ir a “My Apps” e copiar o App Id.

No nosso projecto, ir ao ficheiro strings.xml que se encontra na pasta values e:

```
<string name="facebook_app_id">APP_ID</string>
```

Com a string criada, vamos ao Manifest novamente fazer a ligação com o seguinte código:

```
<meta-data
    android:name="com.facebook.sdk.ApplicationId"
    android:value="@string/facebook_app_id" />
```

Agora que já vimos a dor de cabeça que as permissões dão, vamos prosseguir e testar a aplicação. Vamos correr a aplicação e, verificar os resultados.

Partilhas

Para evitar futuros incidentes e/ou problemas, sempre que iniciarmos a aplicação vamos dizer à mesma para terminar automaticamente sessão ao utilizador. Isto é, caso eu faça login e saia da aplicação, o utilizador autenticado estará sempre autenticado até se realizar logout. Para este exemplo, temos interesse em evitar isso, portanto no método onCreate, antes da inicialização do CallbackManager realizar:

```
LoginManager.getInstance().logOut();
```

Prosseguindo com o desenvolvimento, é do nosso interesse mostrar a todos que eu criei uma aplicação e consigo realizar o login, então vamos mostrar a todos os nossos amigos no facebook o que conseguimos fazer:

```
private void publishOnMyFeed(){
    Profile pt = Profile.getCurrentProfile();
    ShareLinkContent content = new ShareLinkContent.Builder()
            .setContentUrl(pt.getLinkUri())
            .setContentDescription("Olá, consegui entrar no meu próprio
facebook")
            .build();
ShareDialog.show(this, content);

}
```

Esta função, vai buscar o Perfil de utilizador autenticado e, vai partilhar no nosso feed um link (neste caso link do nosso próprio facebook) e, vai escrever uma mensagem.

Chamar esta função no método onSuccess do LoginClick. Correr aplicação.

E agora, erro de que? Permissões… novamente, Android Manifest e colar o seguinte:

```
<provider
    android:name="com.facebook.FacebookContentProvider"

android:authorities="com.facebook.app.FacebookContentProviderAP
P_ID"
    android:exported="true" />
```

Isto, diz que existe um género de “caixa” que nos irá permitir partilhar informação directamente com o nosso facebook. Para mais informação lêr documentação do facebook developers. Correr aplicação!

De seguida, em vez de partilharmos apenas o link para o nosso facebook, vamos partilhar uma imagem, para isso precisam de ter uma imagem na pasta dos Drawable (exemplo do github) e:

```
Bitmap image = BitmapFactory.decodeResource(getResources(), R.drawable.meudrawable);
SharePhoto photo = new SharePhoto.Builder()
        .setBitmap(image)
        .build();

SharePhotoContent photoContent = new SharePhotoContent.Builder()
        .addPhoto(photo)
        .build();

ShareDialog.show(this, photoContent);
```

Informação do utilizador

O FacebookSDK permite, por via de duas formas, obter informação básica do utilizador autenticado. Esta informação, depende do tipo de permissões que temos. A informação básica que vamos pretender extrair vai ser o ID do facebook, o nome e o link direto para o seu facebook e vão ser mostradas as duas vias de o realizar.

Via JSON object

Uma das vias muito útil para obter muita informação é a de obter objectos JSON.

```
{
  "id": "12345678",
  "birthday": "1/1/1950",
  "first_name": "Chris",
  "gender": "male",
  "last_name": "Colm",
  "link": "http://www.facebook.com/12345678",
  "location": {
    "id": "110843418940484",
    "name": "Seattle, Washington"
  },
  "locale": "en_US",
  "name": "Chris Colm",
  "timezone": -8,
  "updated_time": "2010-01-01T16:40:43+0000",
  "verified": true
}
```

Para obtermos os dados que pretendemos teremos que declarar uma variável global:

```
public class MainActivity extends AppCompatActivity {
    private CallbackManager callbackManager;
    private AccessToken accessToken;
```

Sempre que o utilizador se autentica, é guardado um token do utilizador. De seguida, no método onCreate acrescentar a linha:

```
accessToken = AccessToken.getCurrentAccessToken();
```

Após isto, é só criar uma função com o seguinte código:

```
private void getMyInformation(){
    GraphRequest request = GraphRequest.newMeRequest(
            accessToken,
            new GraphRequest.GraphJSONObjectCallback() {
                @Override
                public void onCompleted(
                        JSONObject object,
                        GraphResponse response) {
                    // Application code
                    JSONArray jsonArray = response.getJSONArray();
                    try {
                        String b = object.getString("name");
                        String a = (String)jsonArray.get(1);
                        Log.d("Mensagem", a);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            });
    Bundle parameters = new Bundle();
    parameters.putString("fields", "id,name,link");
    request.setParameters(parameters);
    request.executeAsync();
}
```

Ir ao .xml da main activity e acrescentar (é imperativo que coloquem os nomes iguais para evitar “desgraças”):

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:text="Medium Text"
    android:id="@+id/info_user"
    android:layout_gravity="center_horizontal" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Info"
    android:id="@+id/btnInfo"
    android:layout_gravity="center_horizontal"
    android:onClick="InfoUser"/>
```

Quando clico no botão, acontece o seguinte:

```
public void InfoUser(View view) {
    if(accessToken == null) {
        TextView tv = (TextView)findViewById(R.id.info_user);
        tv.setText("[ERRO] - Access Token Inválido");
        return;
    }

    getMyInformation();
}
```

Se o access token existir (se houver alguém autenticado) ele vai realizar a operação que desejamos, senão irá mostrar erro pois não tem utilizadores autenticados. Para que tudo isto

funcione, é necessário aquando o login do utilizador, guardar o access token. Então, temos de ir ao onSuccess da função LoginClick e acrescentar o seguinte:

```
public void LoginClick(View view) {
    LoginManager.getInstance().logInWithReadPermissions(this,
Arrays.asList("public_profile"));
    LoginManager.getInstance().registerCallback(callbackManager, new
FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {

            //publishOnMyFeed();
            accessToken = AccessToken.getCurrentAccessToken();
        }
(…)
```

A função publishOnMyFeed está comentada para evitar perdas de tempo. Correr a aplicação e testar!

Via Profile Class

Esta é a parte em que o leitor deste manual vai achar menos piada. Tanto trabalho acima realizado e, para obter a mesma informação, bastava realizar o seguinte:

- Criar função getMyInformationByProfile()
- Alterar a função InfoUser() comentando a chamada à função getMyInformation() e chamar a função criada acima.

```
private void getMyInformationByProfile(){
    Profile profile = Profile.getCurrentProfile();
    TextView tv = (TextView)findViewById(R.id.info_user);
    tv.setText("Nome: " + profile.getName() + "\nID: " + profile.getId() +
"\nLink: " + profile.getLinkUri().toString());
}

public void InfoUser(View view) {
    if(accessToken == null) {
        TextView tv = (TextView)findViewById(R.id.info_user);
        tv.setText("[ERRO] - Access Token Inválido");
        return;
    }

    //getMyInformation();
    getMyInformationByProfile();
}
```

Como poderão verificar, o resultado é exactamente o mesmo e, o trabalho é muito menor.

Partilhar diretamente

Este capitulo foi deixado para o fim uma vez que é muito semelhante ao de partilhar fotografias/mensagens no feed em que a única diferença é que este vai partilhar sem pedir ao utilizador para publicar directamente. Vamos ver os resultados acrescentando o seguinte código à função acima criada:

```
private void getMyInformationByProfile(){
    Profile profile = Profile.getCurrentProfile();
    TextView tv = (TextView)findViewById(R.id.info_user);
    tv.setText("Nome: " + profile.getName() + "\nID: " +
profile.getId() + "\nLink: " + profile.getLinkUri().toString());

    ShareLinkContent content = new ShareLinkContent.Builder()
            .setContentUrl(profile.getLinkUri())
            .setContentDescription("Olá, criei uma aplicação brutal
mas não percebi nada do que me explicaram!!!")
            .build();

    ShareApi.share(content, null);
}
```

Correr e verificar o mural do facebook.