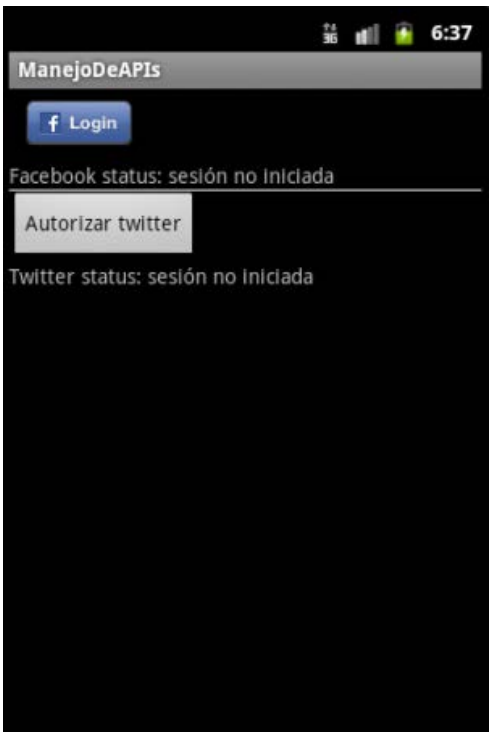
Curso Android: Trabajando con APIs (Facebook y Twitter)

En el capítulo 9 del Curso Android utilizaremos los APIs de Facebook y Twitter para autenticación. La aplicación que realizaremos nos permitirá iniciar sesión y obtener algunos datos.

Queremos que al finalizar se vea así:



Disposición inicial

Para realizar la autenticación es necesario solicitar una aplicación en Facebook y una aplicación en Twitter. En el código completo se incluyen el identificador de la aplicación de Facebook de demo del SDK y un `token/secret` de consumer de una aplicación de Twitter que han sido expirados y si no los reemplazan por los suyos la aplicación no funcionará.

Sobre el Autor

Adrian Catalán
Ingeniero en Sistemas, catedrático universitario, educador y desarrollador de software, fundador del grupo de tecnologías de Google en Guatemala (GTUG), apasionado por aplicaciones web (Ruby On Rails) y móviles (Android) desde el 2011 un Elemental Geek.

Últimos artículos:

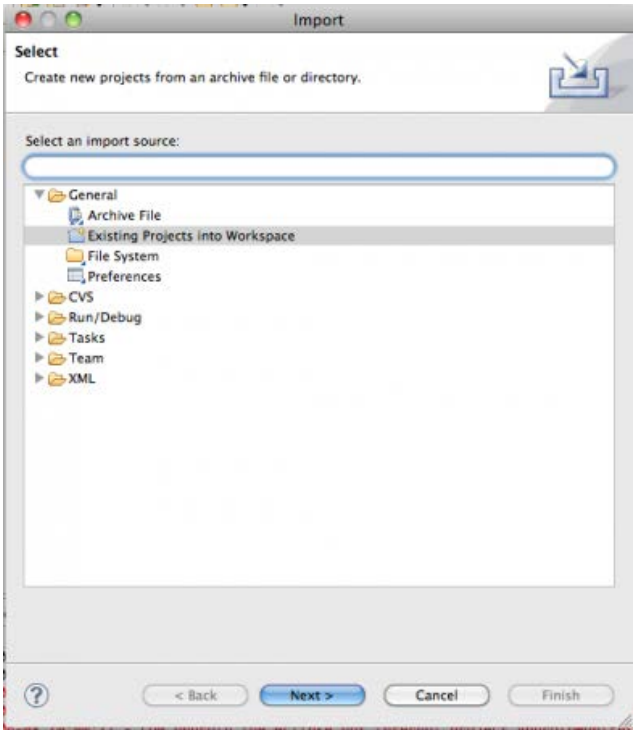
- Curso Android: Conectándonos con APIs de Google
- Curso Android: Envío de emails utilizando Android
- Curso Android: Reproducción de sonido en un ciclo infinito
- Curso Android: Trabajar con el acelerómetro

Más artículos de Adrian Catalán

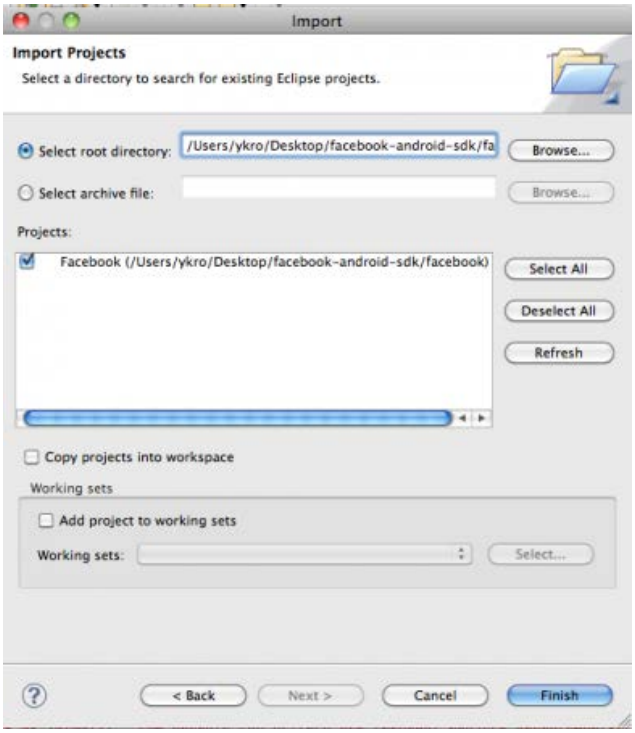
En esta ocasión no utilizaremos código base si no que realizaremos toda la configuración inicial paso a paso. Vamos a utilizar el SDK de Facebook para Android que pueden descargar desde GitHub. Para nuestro ejemplo vamos a tomar no solo el SDK si no también algunos archivos del ejemplo simple que se incluye en el mismo SDK (El botón de inicio de sesión, el manejo y almacenamiento de la sesión y el listener).

El SDK junto a estos archivos del ejemplo simple están unidos en un proyecto que pueden descargar.

Importamos el proyecto a nuestro workspace:



Seleccionando el directorio con los archivos descargados:

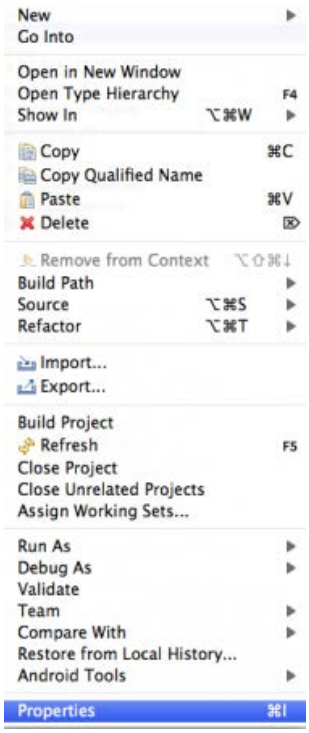


Limitaciones y excepciones al derecho de autor

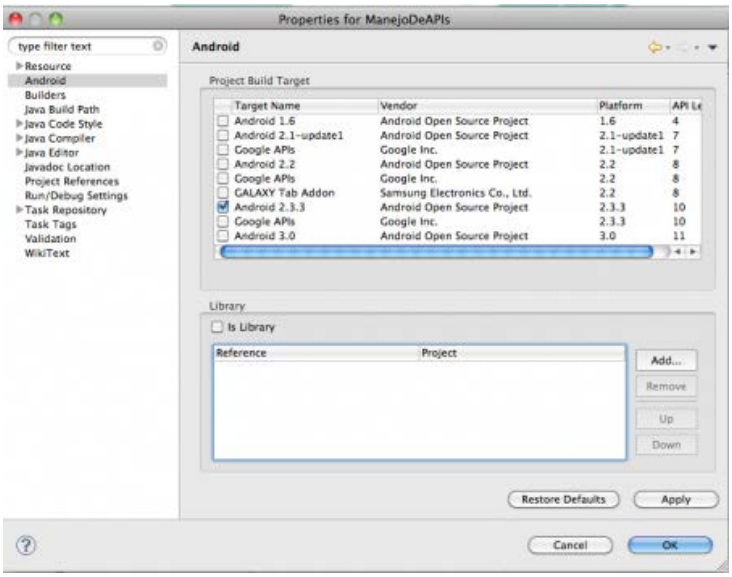
En Argentina y Colombia, curso de desarrollo rápido de apps web con Python y Django

Hacemos un proyecto nuevo, como lo hemos visto en guías anteriores, en mi caso le llamaré **ManejoDeAPIs**.

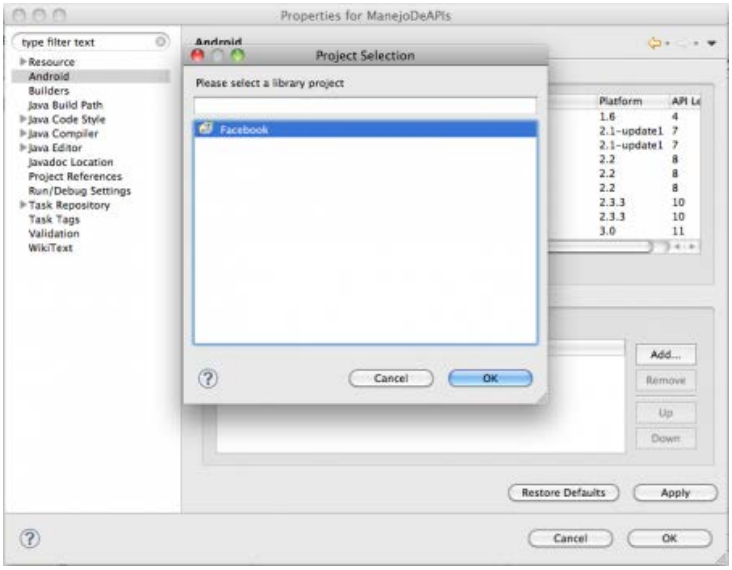
Luego de que está listo el proyecto debemos indicarle que tome la librería del SDK desde otro proyecto, hacemos click derecho sobre el proyecto y luego en propiedades



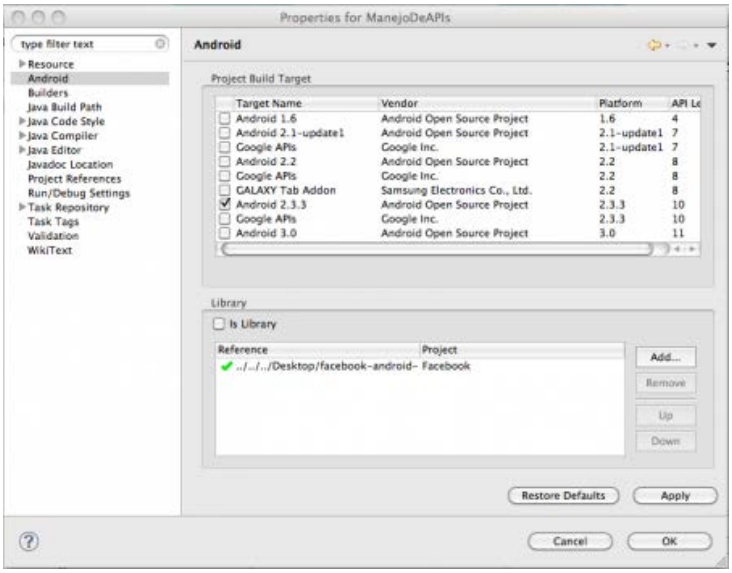
En la pantalla de las opciones buscamos la parte de Android:



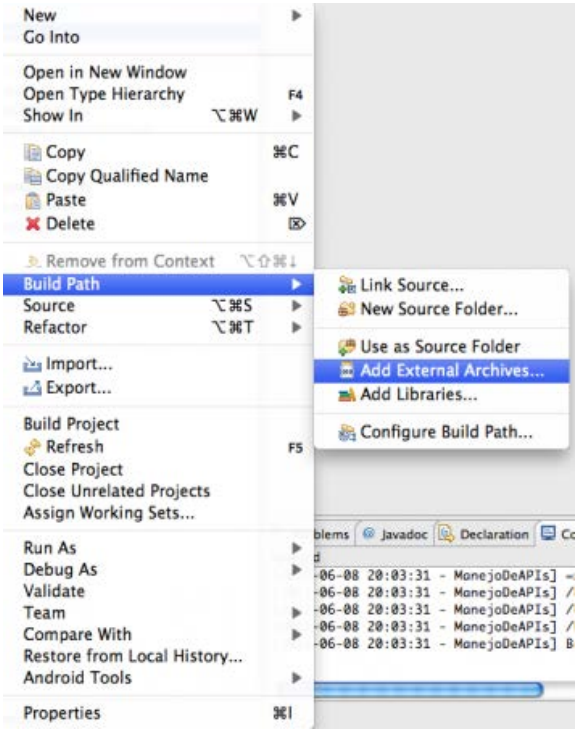
Hacemos click sobre **“Add”** en la parte de **Library**:



Deberá aparecernos el proyecto llamado **"Facebook"**, lo seleccionamos y estará lista la referencia:



Para la autenticación con Twitter hay varias opciones en nuestro caso estaremos utilizando OAuth SignPost y para hacerlo parte de nuestro proyecto necesitamos las librerías de Core y Apache Common, ambos son archivos JAR descargables, la forma de incluirlas es haciendo click derecho sobre el proyecto, luego **"Build Path"** y por último **"Add External Archives"**.



Por último, vamos a configurar algunas cosas en el **Manifest** necesitamos permisos para internet:

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

Para la autenticación de Twitter utilizaremos un `intent filter` que permita que el *callback* luego de la autorización sea nuestra aplicación, para ello usamos el tag de data con un esquema y un host definido por nosotros, es decir, el URL de callback será **mdw://twitter**

```
<intent-filter>
    <action android:name="android.intent.action.VIEW"></action>
    <category android:name="android.intent.category.DEFAULT"></category>
    <category android:name="android.intent.category.BROWSABLE"></category>
    <data android:scheme="mdw" android:host="twitter"></data>
</intent-filter>
```

Diseño

El diseño tendrá 2 botones para **iniciar/cerrar** sesión y 2 etiquetas para mostrar el status. En el caso de Facebook, vamos a utilizar el botón incluido en el ejemplo del SDK implementado en una clase llamada `LoginButton`.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <com.facebook.android.LoginButton
        android:id="@+id/btnFbLogin"
        android:src="@drawable/login_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
    <TextView
        android:layout_height="wrap_content"
        android:text="Facebook status: sesión no iniciada"
        android:id="@+id/txtFbStatus"
        android:layout_width="fill_parent"></TextView>
    <View
        android:layout_height="1dip"
        android:layout_width="fill_parent"
        android:background="#FFFFFF"
    />
    <Button
        android:text="Twitter"
        android:id="@+id/btnTwLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"></Button>
    <TextView
        android:layout_height="wrap_content"
        android:text="Twitter status: sesión no iniciada"
        android:id="@+id/txtTwStatus"
        android:layout_width="fill_parent"></TextView>
</LinearLayout>
```

Agregando código

Inicialmente, vamos a definir algunas variables globales dentro de nuestra `Activity`. El manejo del botón de **autorizar/de-autorizar** de Twitter lo haremos nosotros (el de Facebook tiene una implementación en el código importado que vamos a aprovechar) para ellos lo declaramos como una variable global y declaramos los 2 listeners que utilizaremos.

```
1  private Button btnTwLogin;
2  private OnClickListener twitter_auth, twitter_clearauth;
```

Además requerimos de banderas para chequear el estatus de autenticación y etiquetas para mostrar el mismo.

```
1  private TextView txtFbStatus, txtTwStatus;
2  private boolean twitter_active = false, facebook_active = false;
```

Las peticiones para interactuar con el Graph API de Facebook son asincrónicas y las realizamos a través de una instancia de la clase `AsyncFacebookRunner`.

```
1  private AsyncFacebookRunner mAsyncRunner;
```

También definimos el identificador de la aplicación de Facebook (recuerden cambiarlo por el la aplicación que ustedes solicitaron).

```
1  public static final String APP_ID = "175729095772478";
```

Para la parte de autenticación con Twitter, necesitamos un provider y un consumer configurados con los URLs (no cambian), el consumer key (es necesario cambiarlo por el de ustedes) y el consumer secret (también es necesario cambiarlo por la aplicación que ustedes solicitaron).

```
01  private static CommonsHttpOAuthProvider provider =
02          new CommonsHttpOAuthProvider(
03          "https://api.twitter.com/oauth/request_token",
04          "https://api.twitter.com/oauth/access_token",
05          "https://api.twitter.com/oauth/authorize");
06
07  private static CommonsHttpOAuthConsumer consumer =
08          new CommonsHttpOAuthConsumer(
09              "7iEjG84wItGvXaIZFXAyZg",
10              "sZKCJaUN8BgmYy4r9Z7h1I4BEHV8aAd6Ujw3hofQ4k");
```

Una vez realizada la autorización del usuario con su cuenta de Twitter, recibimos de vuelta un key y un secret de acceso y es necesario guardarlos para cualquier futura interacción.

```
1  private static String ACCESS_KEY = null;
2  private static String ACCESS_SECRET = null;
```

Estas últimas 4 variables son estáticas debido al funcionamiento de la librería OAuth SignPost en algunas ocasiones se pierden los valores luego de autenticar y previo a recibir el acceso entonces puede guardarse temporalmente y luego restaurarse o utilizarse variables estáticas, en este caso hemos tomado la segunda forma de solucionarlo.

Cuando el usuario se autentique con sus credenciales de Facebook, haremos una petición al Graph API para obtener sus datos (identificador y nombre), esta requisición es asincrónica y requiere un `RequestListener` (parte del SDK de Facebook) para realizar alguna acción una vez se ha recibido respuesta.

Vamos a encapsular en un método esta llamada y en concreto nos concentramos en el método `onComplete` del Listener para realizar lo que queremos. La respuesta recibida en JSON es necesario reconocerla (parsing) y luego enviar los valores que nos interesan (ID y nombre) al hilo de ejecución(thread) principal para que modifique la vista(únicamente él, el thread principal, puede hacer este cambio en el contenido del TextView) .

```
01  private void updateFbStatus(){
02      mAsyncRunner.request("me", new RequestListener() {
03          @Override
04          public void onMalformedURLException(MalformedURLException e, Object state) {}
05
06          @Override
07          public void onIOException(IOException e, Object state) {}
```

```
08
09          @Override
10          public void onFileNotFoundException(FileNotFoundException e, Object state) {}
11
12          @Override
13          public void onFacebookError(FacebookError e, Object state) {}
14
15          @Override
16          public void onComplete(String response, Object state) {
17               try {
18                    JSONObject json = Util.parseJson(response);
19                    final String id = json.getString("id");
20                    final String name = json.getString("name");
21                    Main.this.runOnUiThread(new Runnable() {
22
23                        @Override
24                        public void run() {
25                            txtFbStatus.setText("Facebook status: sesión iniciada como " + name + " con el id
26  " + id);
27                        }
28                    });
29                } catch (JSONException e) {
30                    e.printStackTrace();
31                } catch (FacebookError e) {
32                    e.printStackTrace();
33                }
34          }
35      });
36
    }
```

Además de este método, vamos a trabajar en `onCreate`. Primero obtenemos botones y etiquetas (TextView) de status tanto de Facebook como de Twitter.

```
1  LoginButton mLoginButton = (LoginButton) findViewById(R.id.btnFbLogin);
2  btnTwLogin = (Button)findViewById(R.id.btnTwLogin);
3
4  txtTwStatus = (TextView) this.findViewById(R.id.txtTwStatus);
5  txtFbStatus = (TextView) this.findViewById(R.id.txtFbStatus);
```

Luego, construimos nuestro objeto Facebook (parte del SDK descargado) y a partir de él un objeto `AsyncFacebookRunner` que vamos a utilizar para las peticiones al GraphAPI.

```
1  Facebook mFacebook = new Facebook(APP_ID);
2  mAsyncRunner = new AsyncFacebookRunner(mFacebook);
```

Vamos a guardar las credenciales de Twitter como preferencias de la aplicación (funciona como un diccionario, key/value), más adelante vemos la forma de guardarlas al realizar la autenticación, en `onCreate` vamos a recuperarlas y si no existen tenemos un valor por defecto que es un String vacío. Si existen las credenciales entonces habilitamos la variable boolean `twitter_active`.

```
1  SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
2  String stored_keys = prefs.getString("KEY", "");
3  String stored_secret = prefs.getString("SECRET", "");
4
5  if (!stored_keys.equals("") && !stored_secret.equals("")) {
6      twitter_active = true;
7  }
```

Realizamos un proceso similar para el caso de Facebook, aquí es más sencillo porque no es tarea nuestra la administración de las credenciales, nos apoyamos en el código del ejemplo del SDK. Restauramos la sesión y luego validamos la misma, en base a esto habilitamos la bandera `facebook_active` y si la sesión está activa entonces mostraremos los datos del usuario llamando a `updateFbStatus()`.

```
1  SessionStore.restore(mFacebook, this);
2  facebook_active = mFacebook.isSessionValid();
3  if (facebook_active) {
4      updateFbStatus();
5  }
```

Para el manejo del inicio de sesión de Facebook es necesario un Listener para autenticación y otro para cuando se finaliza sesión. Para el primero (AuthListener) es necesario sobrecargar métodos dependiendo si tuvo éxito o no, en caso de tener éxito llamamos a updateFbStatus y en caso de fallar reportamos el error a través del TextView de estatus.

```
01  SessionEvents.addAuthListener(new AuthListener() {
02      @Override
03      public void onAuthSucceed() {
04          updateFbStatus();
05      }
06
07      @Override
08      public void onAuthFail(String error) {
09          txtFbStatus.setText("Facebook status: imposible iniciar sesión " + error);
10      }
11  });
```

En el caso del listener para finalizar la sesión, necesitamos sobrecargar métodos para el

inicio y finalización del proceso de cierre de sesión, le reportaremos al usuario estos eventos a través del TextView de estatus.

```
SessionEvents.addLogoutListener(new LogoutListener() {

@Override
public void onLogoutFinish() {
    txtFbStatus.setText("Facebook status: sesión no iniciada");
}

@Override
public void onLogoutBegin() {
    txtFbStatus.setText("Facebook status: cerrando sesión...");
}
});
```

Finalizamos con la inicialización el botón de login de Facebook:

```
mLoginButton.init(this, mFacebook);
```

Para la parte de Twitter manejaremos 2 Listeners para el evento del click, esta vez vamos a definirlos y asignarles nombre en vez de usar clases internas y anónimas porque los asignaremos varias veces (dependiendo del estado de la autenticación).

El primer listener, el utilizado cuando aun no se inicia sesión, requiere que al darle click al botón obtenga el requestToken y a través de un intent con el URL recibido levante una actividad (el navegador) y muestre al usuario la pantalla de Twitter solicitando su aprobación.

```
twitter_auth = new OnClickListener() {
    @Override
    public void onClick(View v) {
        txtTwStatus.setText("Twitter status: iniciando sesión");

        try {
            String authUrl = provider.retrieveRequestToken(consumer, "mdw://twitter");
            startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(authUrl)));
        } catch (OAuthMessageSignerException e) {
            e.printStackTrace();
        } catch (OAuthNotAuthorizedException e) {
            e.printStackTrace();
        } catch (OAuthExpectationFailedException e) {
            e.printStackTrace();
        } catch (OAuthCommunicationException e) {
            e.printStackTrace();
        }

    }
};
```

El segundo listener, una vez ya ha ocurrido la autenticación, borrará las credenciales guardadas en los SharedPreferences.

```
twitter_clearauth = new OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
        SharedPreferences.Editor editor = prefs.edit();
        editor.putString("KEY", null);
        editor.putString("SECRET", null);
        editor.commit();
        btnTwLogin.setText("Autorizar twitter");
        txtTwStatus.setText("Twitter status: sesión no iniciada ");
        btnTwLogin.setOnClickListener(twitter_auth);

    }
};
```

Es importante notar que las credenciales recibidas de Twitter (key y secret) no expiran y son válidas mientras el usuario no revoque los permisos de la aplicación, lo que hacemos al desautorizar es borrar las credenciales guardadas por lo que la siguiente vez que presionemos el botón solicitaremos nuevas.

Para terminar el método `onCreate` revisamos el valor de twitter_active y actualizamos el texto del botón y del **TextView** de estatus.

```
if (twitter_active) {
    txtTwStatus.setText("Twitter status: sesión iniciada ");
    btnTwLogin.setText("Deautorizar twitter");
    btnTwLogin.setOnClickListener(twitter_clearauth);
} else {
    btnTwLogin.setText("Autorizar twitter");
    btnTwLogin.setOnClickListener(twitter_auth);
}
```

Además del trabajo realizado en **OnCreate**, necesitamos modificar **OnResume**, una vez autorizado el uso de la aplicación de Twitter por parte del usuario, al recibir el redirect al callback nuestra aplicación lo recibirá por el intent-filter colocado previamente y se ejecutará el método OnResume. Para distinguir si la ejecución es por cualquier interrupción de la

aplicación o porque ya nos autenticamos con Twitter revisamos la data del intent y que esta sea un URI de la forma que definimos (**mdw://twitter**), si ese fuera el caso entonces recibimos token y secret de acceso y lo guardamos.

```
provider.retrieveAccessToken(consumer,verifier);
ACCESS_KEY = consumer.getToken();
ACCESS_SECRET = consumer.getTokenSecret();
```

Para almacenar las credenciales de forma persistente utilizaremos SharedPreferences es necesario obtener un editor, guardar la data y luego realizar commit.

```
SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferences.Editor editor = prefs.edit();
editor.putString("KEY", ACCESS_KEY);
editor.putString("SECRET", ACCESS_SECRET);
editor.commit();
```

Luego actualizamos el **TextView** del estado, el texto del botón y la acción asociada al mismo.

```
TextView txtTwStatus = (TextView) this.findViewById(R.id.txtTwStatus);
txtTwStatus.setText("Twitter status: sesión iniciada ");

btnTwLogin.setText("Deautorizar twitter");
btnTwLogin.setOnClickListener(twitter_clearauth);
```

Adicional a esto y fuera de la condición de que el intent tenga data, es posible que el método `onResume` sea llamado y los valores del **TextView** de status de Facebook se hayan perdido entonces es necesario revisarlo.

```
if (facebook_active) {
    updateFbStatus();
}
```

El método completo `onResume` queda de la siguiente forma:

```
@Override
  public void onResume() {
    super.onResume();
    Uri uri = this.getIntent().getData();
    if (uri != null && uri.toString().startsWith("mdw://twitter")) {
        String verifier = uri.getQueryParameter(OAuth.OAUTH_VERIFIER);
        try {
            provider.retrieveAccessToken(consumer,verifier);
            ACCESS_KEY = consumer.getToken();
            ACCESS_SECRET = consumer.getTokenSecret();

            SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("KEY", ACCESS_KEY);
            editor.putString("SECRET", ACCESS_SECRET);
            editor.commit();

            TextView txtTwStatus = (TextView) this.findViewById(R.id.txtTwStatus);
                txtTwStatus.setText("Twitter status: sesión iniciada ");

                btnTwLogin.setText("Deautorizar twitter");
                btnTwLogin.setOnClickListener(twitter_clearauth);

            } catch (OAuthMessageSignerException e) {
                e.printStackTrace();
            } catch (OAuthNotAuthorizedException e) {
                e.printStackTrace();
            } catch (OAuthExpectationFailedException e) {
                e.printStackTrace();
            } catch (OAuthCommunicationException e) {
                e.printStackTrace();
            }

    }

      if (facebook_active) {
        updateFbStatus();
      }
  }
```

Descargar:

Puedes descargar el código de la aplicación completa y funcional en (Github): Trabajando con APIs (Facebook y Twitter).

Conclusión

En este capítulo hemos visto varias cosas:

- Utilización de un proyecto como librería (SDK de Facebook para Android), es posible crear un proyecto sin Activities si no únicamente con código para ser utilizado por terceros. En este caso, nosotros aprovechamos este código creado por alguien más para

nuestra aplicación.

- Uso de librerías externas a través de archivos JAR que se vuelven parte del proyecto, en nuestra aplicación de demo de esta forma incluimos OAuth SignPost.
- Conexión con APIs de terceros (Facebook & Twitter) para lograr la autenticación, vimos dos opciones que podemos manejar para poder identificar a los usuarios de nuestras aplicaciones. La librería de OAuth SignPost es posible utilizarla también con algunos otros proveedores que trabajan también con OAuth.

Siguiente capítulo: Conectándonos con APIs de Google →

Capitulos de la Guía Android

Adrian Catalán para Maestros del Web.
Agrega tu comentario | Enlace permanente al artículo

Síguenos en:

@maestros | Fan page

2 Comentarios

Pedro Diaz 1

Excelente Turorial, muy explicativo ha sido de gran ayuda. Muchas Gracias!. Esperamos seguir viendo mas tutoriales de Android de Maestros del web y Adrian Catalán. Felicidades

11.06.2011 - 15:47

Juan 2

Muy buen tutorial!

Tengo una problema…

cuando corro el programa en el emulador no hay problema y se logea de manera perfecta..

el tema es que cuando uso este ejemplo en un telefono con la aplicacion de Facebook instalada no recibe los datos de token ni de nada, porque los recupera de una pagina web y al abrise la aplicacion esta no retorna nada para que se actualice el boton y me de el ok de logeado

desintale la app de facebook en el telefono y volvi a probar el ejemplo y funciona. xq abre la version movil de facebook y esta al logearse y conceder el acceso devuelve los datos q usa el ejemplo…

la pregunta es si puedo forzar al ejemplo a abrir la version movil de facebook y no la app

no se si me explique bien..

en fin, muchas gracias!

23.07.2011 - 08:00

Los comentarios de este post están cerrados. Si quieres seguir la discusión, debatir, criticar, sugerir o expandir el tema te invitamos a hacerlo en tu propio blog, en twitter o donde puedas publicar. No olvides enlazar a este post para que sigamos la conversación y se genere un trackback.

1 Trackbacks

Trabajando con apis facebook y twitter en android | Manuales gratis

← Josh Jones y Dallas Kashuba fundadores Dreamhost

Geolocalización, móviles y mapas →