2-4 Android端末での実行

2-4-1 Android端末での実行方法

作成したアプリをAndroid端末で実行する方法には、次の3種類の方法があります。

- USB経由によるデバッグ版アプリの実行
- USB経由によるリリース版アプリの実行
- Android Market経由によるリリース版アプリの実行

デバッグ版アプリはデバッグ用の署名が自動的に付加されて動作しますが、Android Marketに公開することはできません。

Android Marketで配布するリリース版には、アプリに署名を付加する必要があります。署名は証明機関発行のものは必須ではなく、自己証明も利用可能です。開発はデバッグ版アプリで行い、公開前にリリース版アプリで動作テスト後、Android Marketに公開するという流れになります。

2-4-2 パソコンとAndroid端末のUSB接続

パソコンで開発したアプリをAndroid端末で実行するために、パソコンとAndroid端末をUSBで接続します。

① Android端末のMENUボタンを押し、「設定→アプリケーション→開発（Settings→Applications→Development）」を選択する。
②「USBデバッグ（USB debugging）」をチェックする。
③ Android端末をUSBでパソコンと接続する。
④ Windowsではドライバを求めてくるので、Android SDKの「usb_driver」フォルダ内のドライバを指定する。ドライバを自動的にインストールしようとして失敗したり、インストールを求めてこないときは、デバイスマネージャを使って手動でドライバを更新すること。なお、Macではドライバは必要ない。

また、機種によってはSDKにドライバが含まれていない場合がある。たとえば、Xpreiaの場合は、パソコンとXpreiaをUSBメモリとして見えるように接続し、本体の「CDBrowser」フォルダ内にある「Drivers.zip」をコピーして解凍しドライバを取り出す。

⑤ コマンドプロンプトで「adb devices」と入力し、パソコンと接続中のAndroid端末一覧（エミュレータを含む）を表示して、接続していることを確認する。

```
>adb devices
List of devices attached
HT845GZ49615 device
```

2-4-3 USB経由によるデバッグ版アプリの実行

パソコンとAndroid端末のUSB接続が確認できたら、デバッグ版アプリを以下のように実行します。

① Eclipseのメニューから「Run→Debug Configurations」を選択する。
② 左側のツリーの「Android Application」を選択する。
③ 一番左の「New launch configuration」ボタンを押し、ツリーに「New_configuration」を追加する。
④ 画面の右側のNameとProjectに「HelloWorld」と記述する。
⑤「Target」タブを選択する。

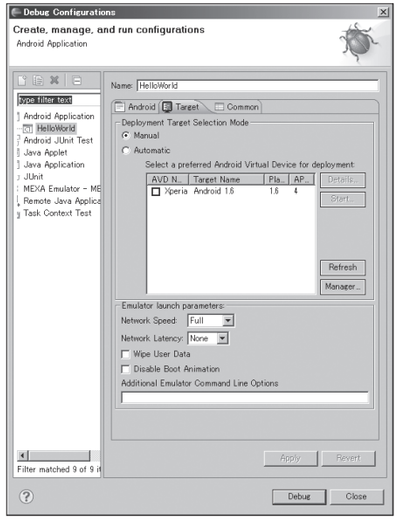

図2-4-1 デバッグ版アプリのための実行設定

⑥「Device Target Selection Mode」で「Manual」を選択する。
⑦「Debug」ボタンを押す。

9-3 Twitterクライアント

9-3-1 Twitterクライアントのプログラムの構成

Twitterクライアントを作ります。初回起動時にTwitter認証画面が表示されます。アカウント名とパスワードを入力することで、Twitterのタイムラインを表示することができます。

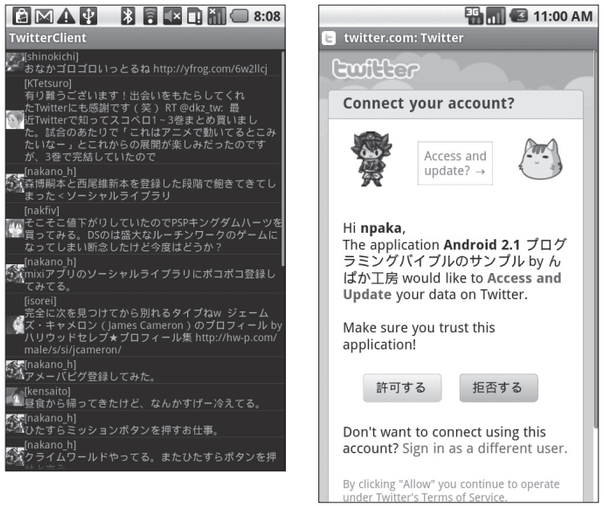

図9-3-1　Twitterクライアントの画面

このプログラムは、次の2つのクラスで構成されています。プロジェクトは、プロジェクト名「OAuth TwitterClient」で作成してください。

- OAuthTwitterClientクラス（OAuthTwitterClient.java）
- Statusクラス（Status.java）

9-3-2 OAuth認証の利用のための登録申請

Twitterは、2010年6月までは認証プロトコルとしてBASIC認証が利用可能でしたが現在は使用できません。利用できるのはOAuth認証とXAuth認証のみとなります。このプログラムでは、OAuth認証を利用します。

OAuth認証を利用するには、まず以下のWebページでTwitterのOAuth認証を利用するための登録申請を行う必要があります。

● TwitterのOAuth認証の登録申請
http://twitter.com/oauth_clients

アプリケーションの種類として「ブラウザアプリ」、コールバックURLには適当なURL（設定は必須ですがプログラム内では利用されません）を記述してください。登録が完了するとコンシューマーキー（Consumer key）とコンシューマーシークレット（Consumer secret）が取得できるので、それをソースコードの定数の値に記述します。

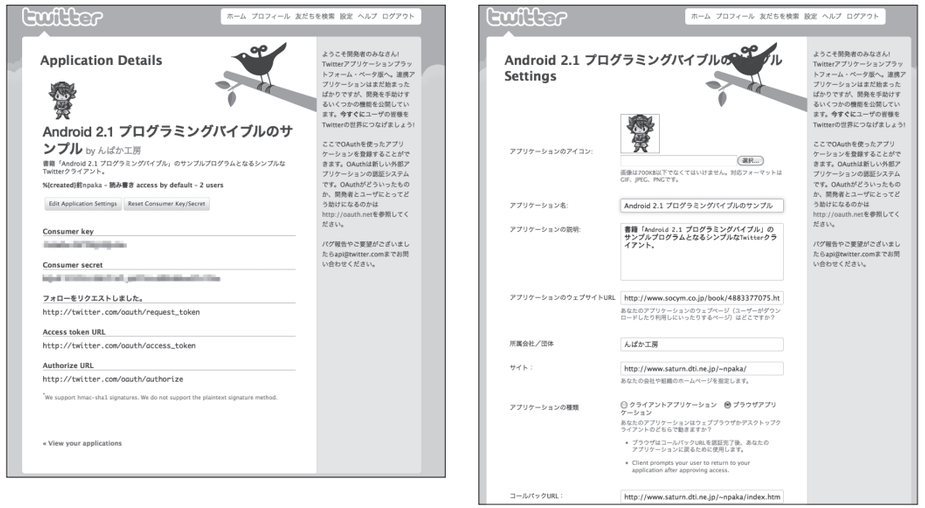

図9-3-2　OAuth認証の利用のための登録申請の画面

9-3-3 OAuthライブラリの追加

OAuth認証の処理には、オープンソースライブラリ「oauth-signpost」を使用しますので、以下の準備をしておきます。

① 以下のWebサイトから「signpost-core-1.2.1.1.jar」と「signpost-commonshttp4-1.2.1.1.jar」をダウンロードする。

● OAuth認証のオープンソースライブラリのダウンロード
http://code.google.com/p/oauth-signpost/

② 「Package Explorerのプロジェクト名を右クリック→New→Folder」を選択して、「lib」という名前のフォルダを作成する。

③ プロジェクトのlibフォルダに「signpost-core-1.2.1.1.jar」と「signpost-commonshttp4-1.2.1.1.jar」をドラッグ＆ドロップでコピーする。

④ 「Package Explorerのプロジェクト名を右クリック→Properties→Java Build Path→Libraries」で「Add JARs」ボタンを押し、「signpost-core-1.2.1.1.jar」と「signpost-commonshttp4-1.2.1.1.jar」を追加する。

⑤ 画面右上部の「Order and Export」を押し、「signpost-core-1.2.1.1.jar」と「signpost-commonshttp4-1.2.1.1.jar」をチェックし、「OK」ボタンを押す。

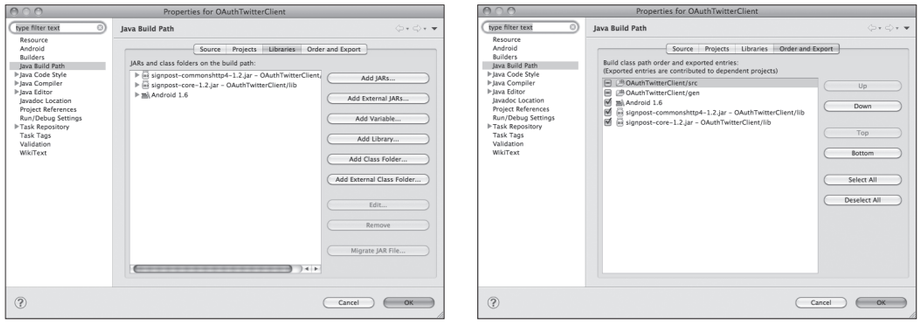

図9-3-3　OAuth認証のオープンソースライブラリの追加

9-3-4　ソースコード

■OAuthTwitterClientクラス

OAuthTwitterClientクラスは、プログラムの本体となるクラスです。

リスト9-3-1　OAuthTwitterClient.java

```
package net.npaka.oauthtwitterclient;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Bundle;
import android.util.Xml;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import oauth.signpost.OAuthProvider;
import oauth.signpost.basic.DefaultOAuthProvider;
import oauth.signpost.commonshttp.CommonsHttpOAuthConsumer;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.xmlpull.v1.XmlPullParser;

//Twitterクライアント
public class OAuthTwitterClient extends Activity {
    //定数
    private final static String
        CONSUMER_KEY   ="コンシューマーキーを記述",
        CONSUMER_SECRET="コンシューマーシークレットを記述";
    private final String CALLBACKURL="myapp://mainactivity";
    private static final int
        MENU_SETUP =0,
        MENU_UPDATE=1;

    //認証
    private CommonsHttpOAuthConsumer consumer;
    private OAuthProvider            provider;

    //情報
    private ListView listView;
    private static HashMap<String,Drawable> icons=
        new HashMap<String,Drawable>();
    private ArrayList<Status> timeline=
        new ArrayList<Status>();

    //アプリの初期化
    @Override
```

```
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        //レイアウトの生成
        LinearLayout layout=new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);
        setContentView(layout);

        //リストビューの生成（1）
        listView=new ListView(this);
        setLLParams(listView);
        layout.addView(listView);

        //認証
        doOauth(false);
    }

    //オプションメニューの生成
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        MenuItem item0=menu.add(0,MENU_SETUP,0,"設定");
        item0.setIcon(android.R.drawable.ic_menu_add);
        MenuItem item1=menu.add(0,MENU_UPDATE,0,"更新");
        item1.setIcon(android.R.drawable.ic_menu_call);
        return true;
    }

    //メニューアイテム選択イベントの処理
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        try {
            switch (item.getItemId()) {
            //設定
            case MENU_SETUP:
                doOauth(true);
                return true;
            //更新
            case MENU_UPDATE:
                updateTimeline();
                return true;
            }
        } catch (Exception e) {
        }
        return true;
    }

    //タイムラインの更新
    private void updateTimeline() {
        //ArrayList<Status>
        timeline=new ArrayList<Status>();
        Status status =null;
        String tagName=null;
```

```
        InputStream in=null;
        try {
            //接続
            HttpGet httpGet=new HttpGet(
                "http://twitter.com/statuses/friends_timeline.xml");
            consumer.sign(httpGet);
            DefaultHttpClient http=new DefaultHttpClient();
            HttpResponse execute=http.execute(httpGet);

            //XMLのパース処理（4）
            in=execute.getEntity().getContent();
            final XmlPullParser parser=Xml.newPullParser();
            parser.setInput(new InputStreamReader(in));
            while (true) {
                int type=parser.next();
                //ドキュメント開始
                if (type==XmlPullParser.START_DOCUMENT) {
                }
                //タグ開始
                else if (type==XmlPullParser.START_TAG) {
                    tagName=parser.getName();
                    if (tagName.equals("status")) {
                        status=new Status();
                        timeline.add(status);
                    }
                }
                //テキスト
                else if (type==XmlPullParser.TEXT) {
                    if (parser.getText().trim().length()==0) {
                    } else if (tagName.equals("screen_name")) {
                        status.name=parser.getText();
                    } else if (tagName.equals("text")) {
                        status.text=parser.getText();
                    } else if (tagName.equals("profile_image_url")) {
                        status.iconURL=parser.getText();
                    }
                }
                //タグ終了
                else if (type==XmlPullParser.END_TAG) {
                }
                //ドキュメント終了
                else if (type==XmlPullParser.END_DOCUMENT) {
                    break;
                }
            }

            //切断
            in.close();

            //更新
            for (int i=0;i<timeline.size();i++) {
                timeline.get(i).icon=readIcon(timeline.get(i).iconURL);
            }
```

```
            listView.setAdapter(new TwitterAdapter(this));
        } catch (Exception e) {
            Toast.makeText(this, e.getMessage(),Toast.LENGTH_LONG).show();
        }
    }

    //アイコンの読み込み
    private Drawable readIcon(String url) throws Exception {
        Drawable drawable=icons.get(url);
        if (drawable!=null) return drawable;
        byte[] data=http2data(url);
        Bitmap bmp=BitmapFactory.decodeByteArray(data,0,data.length);
        drawable=new BitmapDrawable(bmp);
        icons.put(url,drawable);
        return drawable;
    }

    //HTTP通信
    private byte[] http2data(String path) throws Exception {
        int size;
        byte[] w=new byte[1024];
        HttpURLConnection c=null;
        InputStream in=null;
        ByteArrayOutputStream out=null;
        try {
            //HTTP接続のオープン
            URL url=new URL(path);
            c=(HttpURLConnection)url.openConnection();
            c.setRequestMethod("GET");
            c.connect();
            in=c.getInputStream();

            //バイト配列の読み込み
            out=new ByteArrayOutputStream();
            while (true) {
                size=in.read(w);
                if (size<=0) break;
                out.write(w,0,size);
            }
            out.close();

            //HTTP接続のクローズ
            in.close();
            c.disconnect();
            return out.toByteArray();
        } catch (Exception e) {
            try {
                if (c!=null) c.disconnect();
                if (in!=null) in.close();
                if (out!=null) out.close();
            } catch (Exception e2) {
            }
            throw e;
```



```
        }
    }

    //ライナーレイアウトのパラメータ指定
    private static void setLLParams(View view) {
        view.setLayoutParams(new LinearLayout.LayoutParams(
            LinearLayout.LayoutParams.FILL_PARENT,
            LinearLayout.LayoutParams.WRAP_CONTENT));
    }

    //Twitterアダプタの生成（2）
    public class TwitterAdapter extends BaseAdapter {
        private Context context;

        //コンストラクタ
        public TwitterAdapter(Context c) {
            context=c;
        }

        //項目数の取得
        public int getCount() {
            return timeline.size();
        }

        //項目の取得
        public Object getItem(int position) {
            return timeline.get(position);
        }

        //項目IDの取得
        public long getItemId(int position) {
            return position;
        }

        //ビューの取得
        public View getView(int position,
            View convertView,ViewGroup parent) {
            TextView textView=new TextView(context);
            textView.setTextSize(12.0f);
            textView.setCompoundDrawablesWithIntrinsicBounds(
                timeline.get(position).icon,null,null,null);
            textView.setText("["+timeline.get(position).name+"]¥n"+
                timeline.get(position).text);
            return textView;
        }
    }

    //OAuth認証による通信（3）
    private void doOauth(boolean setup) {
        try {
            consumer=new CommonsHttpOAuthConsumer(
                CONSUMER_KEY,CONSUMER_SECRET);
            provider=new DefaultOAuthProvider(
```

```
                "http://twitter.com/oauth/request_token",
                "http://twitter.com/oauth/access_token",
                "http://twitter.com/oauth/authorize");

            //トークンの読み込み
            SharedPreferences pref=getSharedPreferences(
                "token",MODE_PRIVATE);
            String token      =pref.getString("token","");
            String tokenSecret=pref.getString("tokenSecret","");

            //認証済み
            if (!setup && token.length()>0 && tokenSecret.length()>0) {
                consumer.setTokenWithSecret(token,tokenSecret);

                //Twitter操作
                updateTimeline();
            }
            //認証処理のためブラウザ起動
            else {
                String url=provider.retrieveRequestToken(
                    consumer,CALLBACKURL);
                this.startActivity(
                    new Intent(Intent.ACTION_VIEW, Uri.parse(url)));
            }
        } catch (Exception e) {
            Toast.makeText(this,e.getMessage(),Toast.LENGTH_LONG).show();
        }
    }

    //認証完了時に呼ばれる
    @Override
    protected void onNewIntent(Intent intent) {
        super.onNewIntent(intent);
        Uri uri=intent.getData();
        if (uri!=null && uri.toString().startsWith(CALLBACKURL)) {
            String verifier=uri.getQueryParameter(
                oauth.signpost.OAuth.OAUTH_VERIFIER);
            try {
                provider.retrieveAccessToken(consumer,verifier);

                //トークンの書き込み
                SharedPreferences pref=getSharedPreferences(
                    "token",MODE_PRIVATE);
                SharedPreferences.Editor editor=pref.edit();
                editor.putString("token",consumer.getToken());
                editor.putString("tokenSecret",consumer.getTokenSecret());
                editor.commit();

                //Twitter操作
                updateTimeline();
            } catch(Exception e){
                Toast.makeText(this, e.getMessage(),Toast.LENGTH_LONG).show();
            }
```



```
        }
    }
}
```

■Statusクラス

Statusクラスは、Twitterの1つの状態情報を保持するデータ型です。名前、テキスト、アイコンURL、アイコンといった情報を保持します。

リスト9-3-2　Status.java

```
package net.npaka.twitterclient;
import android.graphics.drawable.Drawable;

//ステータス
public class Status {
    public String   name;   //名前
    public String   text;   //テキスト
    public String   iconURL;//アイコンURL
    public Drawable icon;   //アイコン
}
```

9-3-5　ソースコードの解説

プログラムの本体となるOAuthTwitterClient.javaを解説します。Twitterの1つの状態情報を保持するStatus.javaについては解説しません。

(1) リストビューの生成

このプログラムでは、リスト形式で項目を表示する「リストビュー」を利用します。リストビューを生成するには、ListViewクラスを使います。

ListViewクラス

ListView(Context *context*)
機能：ListViewクラスのコンストラクタ
引数：*context*　コンテキスト

フォーカスを指定するにはsetFocusableInTouchMode()メソッド、項目クリック時のイベントを取得するにはsetOnItemClickListener()メソッドを使います。

(2) Twitterアダプタの生成

リストビュー内に表示する情報は、BaseAdapterクラスを継承したクラスのオブジェクトを生成し、それをリストビューに設定します。BaseAdapterクラスを継承するクラスは、以下の4つのメソッドをオーバーライドする必要があります。

BaseAdapterクラス

```
int getCount()
```
機能：リストの項目数の取得
戻り値：リストの項目数

BaseAdapterクラス

```
Object getItem(int position)
```
機能：リストの項目の取得
引数：*position*　リストの項目の位置
戻り値：リストの項目

BaseAdapterクラス

```
Object getItemId(int position)
```
機能：リストの項目IDの取得
引数：*position*　リストの項目IDの位置
戻り値：項目ID

BaseAdapterクラス

```
View getView(int position, View convertView, ViewGroup parent)
```
機能：リストの項目のビューの取得
引数：*position*　　リストの項目の位置
　　contentView　リストのビュー
　　parent　　リストのビューグループ
戻り値：リストの項目のビュー

今回はリストのビューとして、TextViewクラスのsetCompoundDrawablesWithIntrinsicBounds()メソッドで画像を左に追加したものを戻しています。

TextViewクラス

```
void setCompoundDrawablesWithIntrinsicBounds(Drawable left, Drawable
top, Drawable right, Drawable bottom)
```
機能：描画オブジェクトの追加
引数：*left*　左に追加する描画オブジェクト
　　top　上に追加する描画オブジェクト
　　right　右に追加する描画オブジェクト
　　bottom　下に追加する描画オブジェクト

アダプタをリストビューに指定するには、setAdapter()メソッドを使います。

ListViewクラス

```
void setAdapter(ListAdapter adapter)
```
機能：アダプタの指定
引数：*adapter*　アダプタ

(3) OAuth認証による通信

はじめに、CommonsHttpOAuthConsumerクラスでコンシューマを操作するオブジェクトを生成します。コンシューマは、OAuth認証を使ってユーザーデータの提供を受ける側のオブジェクトです。コンストラクタの引数に、アプリ登録申請時に取得した「コンシューマーキー」と「コンシューマーシークレット」を指定します。

CommonsHttpOAuthConsumerクラス

```
CommonsHttpOAuthConsumer(String consumerKey, String consumerSecret)
```
機能：CommonsHttpOAuthConsumerクラスのコンストラクタ
引数：*consumerKey*　　コンシューマーキーを指定
　　consumerSecret　コンシューマーシークレットを指定

次に、DefaultOAuthProviderクラスでサービスプロバイダを操作するオブジェクトを生成します。サービスプロバイダは、OAuth認証を使ってユーザーデータを提供する側のオブジェクトです。コンストラクタにTwitterとOAuth認証を行うためのURLを指定します。

DefaultOAuthProviderクラス

```
DefaultOAuthProvider(String requestTokenEndpointUrl, String
accessTokenEndpointUrl, String authorizationWebsiteUrl)
```
機能：DefaultOAuthProviderクラスのコンストラクタ
引数：*requestTokenEndpointUrl*　「http://twitter.com/oauth/request_token」を指定
　　accessTokenEndpointUrl　「http://twitter.com/oauth/access_token」を指定
　　authorizationWebsiteUrl　「http://twitter.com/oauth/authorize」を指定

続いて、OAuthProviderオブジェクトのretrieveRequestToken()メソッドで認証用ページのURLを取得してブラウザで開きます。コールバック文字列には"myapp://mainactivity"を指定します。

OAuthProviderクラス

```
String retrieveRequestToken(CommonsHttpOAuthConsumer consumer, String
callback)
```
機能：認証用ページのURLを取得
引数：*consumer*　コンシューマ
　　callback　コールバック文字列
戻り値：認証ページのURL

認証用ページでは、Twitterのアカウント名とパスワードを入力します。ブラウザでの認証が完了すると、元のアクティビティに戻りonNewIntent()メソッドが呼ばれます。インテントのデータがコールバック文字列かどうかを調べ、コールバック文字列だった場合は、Uriクラスのget QueryParameter(oauth.signpost.OAuth.OAUTH_VERIFIER)で証明書となるデータを取得します。それをOAuthProviderオブジェクトのretrieveAccessToken()メソッドに代入します。これで認証完了です。

CommonsHttpOAuthConsumerのgetToken()メソッドでトークン、getTokenSecret()メソッドでトークンシークレットを取得できます。この2つの情報を保存しておき、次回アプリ起動時にCommonsHttpOAuthConsumerオブジェクトのsetTokenWithSecret()メソッドでこれらを代入することで、アカウント名とパスワードの入力なしにTwitter APIを利用できるようになります。

CommonsHttpOAuthConsumerクラス

```
void setTokenWithSecret(String token, String tokenSecret)
```
機能：コンシューマにトークンとトークンシークレットを指定
引数：*token*　トークンを指定
　　tokenSecret　トークンシークレットを指定

(4) XMLのパース処理

Twitterサーバとの通信でXMLによる情報が取得できるので、これをXMLパーサーを使って分解し、Statusクラスの配列に変換します。

XMLパーサーを利用するには、Xmlクラスのnew PullParser()メソッドでXmlPullParserオブジェクトを取得し、setInput()メソッドで入力ストリームを指定します。

Xmlクラス

```
static XmlPullParser newPullParser()
```
機能：XMLパーサーの取得
戻り値：XMLパーサー

XmlPullParserクラス

```
void setInput(Reader in)
```
機能：入力ストリームの指定
引数：*in*　入力ストリーム

そして、XmlPullParserオブジェクトのnext()メソッドで次のタグ種別を、getName()メソッドで名前を、getText()メソッドでテキストを取得できます。

XmlPullParserクラス

```
int next()
```
機能：次のタグ種別の取得
戻り値：タグ種別

タグ種別としては、次の定数が戻ります。

タグ種別の定数	意味
XmlPullParser.START_DOCUMENT	ドキュメント開始
XmlPullParser.START_TAG	タグ開始
XmlPullParser.TEXT	テキスト
XmlPullParser.END_TAG	タグ終了
XmlPullParser.END_DOCUMENT	ドキュメント終了

表9-3-1　next()メソッドで戻るタグ種別

XmlPullParserクラス

```
String getName()
```
機能：タグの名前の取得
戻り値：タグの名前

XmlPullParserクラス

```
String getText()
```
機能：タグのテキストの取得
戻り値：タグのテキスト

9-3-6　AndroidManifest.xmlへのパーミッションとインテントフィルタの指定

今回は通信機能を使うので、AndroidManifest.xml の「User Permission」の「android.permission.INTERNET」を追加します。さらに、認証処理を行うブラウザからのアクセストークンを受け取るので、インテントフィルタを設定します。さらに、アクティビティにandroid:launchMode="singleInstance"も指定します。

結果として、AndroidManifest.xmlに以下のようにuses-permissionタグとintent-filterタグが追加されていることを確認してください。

リスト9-3-3　AndroidManifest.xmlの記述

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.npaka.oauthtwitterclient"
    android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">

        <activity
            android:name=".OAuthTwitterClient"
            android:label="@string/app_name"
            android:launchMode="singleInstance">
            <intent-filter>
```

```
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="myapp" android:host="mainactivity" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="4" />

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

</manifest>
```