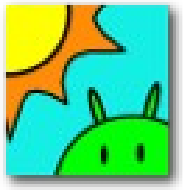
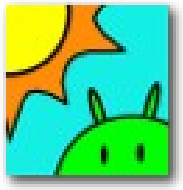


日本Androidの会 関西支部
『初心者向け勉強会』

「今日の天気」アプリを
つくってみました

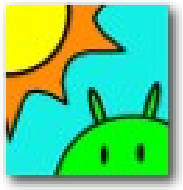


自己紹介

・名前　　：robo　（兼高理恵）

・お仕事　：Java技術者

・好きな物：モバイル端末



アプリ作成のまえに

Android (基礎)についてのおさらい

アプリを作るには、

Android の基礎知識が必要です…ね

まずは、Android(基礎/概念)について、

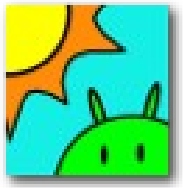
おさらいしてみます。

アプリ作成のまえに

Android で出来ること

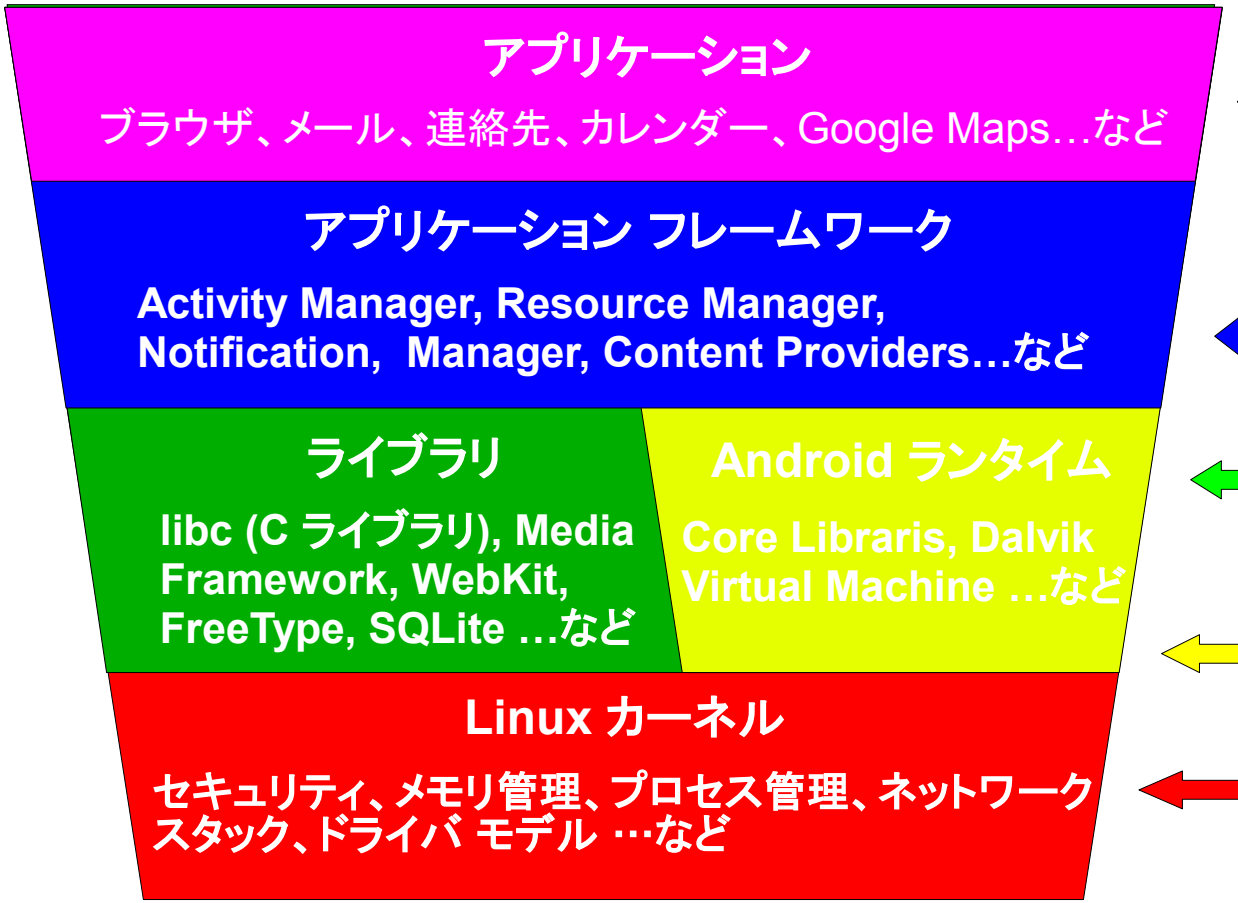
- WebKit ベースのWebブラウジング環境
- 2D/3D グラフィックスや、SQLite によるデータ管理
- 音声/映像/静止画各種メディア・フォーマットの取り扱い
  (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIFなど)
- Bluetooth、EDGE、3G、WiFi、カメラ、GPS、コンパス、加速度計
  (※)ハードウェア依存
- 無料(無償)提供開発環境による自由なアプリケーション開発
  (デバッガやエミュレータにEclipseプラグインなどが提供されています)
- 自由に開発したアプリケーションの自由な配布
  (Android Marketでの提供、勝手アプリとしての配信も可)

参考URL
android developers / Androidの基本 > Androidとは > 特長
http://developer.android.com/intl/ja/guide/basics/what-is-android.html



アプリ作成のまえに

Androidのアーキテクチャ



アプリケーションは、
Java プログラミング言語で作成されます。

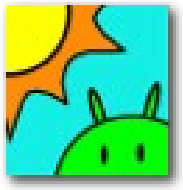
アプリケーション構築用に公開(提供)された、
Android システム各種機能のコンポーネント。
プログラムからは、Framework APIを介して
利用されます。

Android システム各種コンポーネントで
使用される、C/C++ ライブラリのセット。
ライブラリは、Framework API から利用されます。

Java 言語のコア・ライブラリと、
プロセス、スレッディング、メモリ管理という
プログラム実行基盤機能を Linux カーネルを
使用して提供する Dalvik VM を含みます。

コア システム サービスを提供します。

参考URL
android developers / Androidの基本 > Androidとは > Android アーキテクチャ
http://developer.android.com/intl/ja/guide/basics/what-is-android.html



アプリ作成のまえに

Android アプリのセキュリティ

Android アプリケーションは、

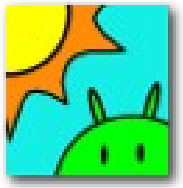
一般的に他アプリのプログラムやデータにアクセスできません。

ただし、アプリ相互の設定で可能にすることもできます。

Android は、Linux 機能を利用して、
アプリごとに個別の Linux ユーザ ID と Linux プロセスを割り当て、
個別の Linux プロセスごとに Java 仮想マシンを起動してアプリを実行させて、
アプリごとの独立性を確保しています。

他のアプリとプログラムやデータを共有するには、AndroidManifest.xml で
Linux ユーザ ID や Linux プロセスを共有する設定をしなくてはなりません。

参考URL
android developers / フレームワークトピック > 開発の基礎
http://developer.android.com/intl/ja/guide/topics/fundamentals.html



アプリ作成のまえに

Android アプリのコンポーネント

(Android アプリケーションの実行と連携の基盤)

Android アプリケーションは、

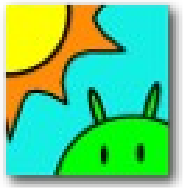
他アプリを直接利用することができませんが、

許可(公開)されている、他アプリの要素なら間接的に利用できます。

Android では、画面を通じたユーザとの対話や、要求に即した自動処理など、
アプリケーションに必須の各種応答機能の基盤が
Android システムから生成/実行可能な ***コンポーネント*** として提供されています。

Android では、これらコンポーネントを利用したアプリ・プログラムの構築と、
AndroidManifest.xml での利用コンポーネントの宣言や使用条件の設定を前提に、
Intent や URI 指定による Android システムを介した各種基礎応答機能の起動や
呼出(他アプリ要素の間接利用)を可能にしています。

参考URL
android developers / フレームワークトピック > 開発の基礎 > アプリケーションのコンポーネント
http://developer.android.com/intl/ja/guide/topics/fundamentals.html



アプリ作成のまえに

Android アプリのコンポーネント

(Android システム提供の4つの応答機能基盤)

アクティビティ(Activity)

自アプリにおける ***画面を通じたユーザとの*** 対話処理を担います。
画面には、ボタンやテキストフィールドなどの各種画面パーツが配置可能です。
画面表示構成は、各画面パーツ(View)と、それらを子要素に持ち配置を指定する
レイアウト(ViewGroup)との入れ子構造により指定されます。

サービス(Service)

自アプリにおける ***画面を必要としない*** バックグラウンド処理を担います。

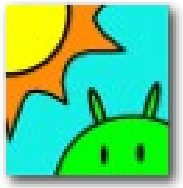
ブロードキャスト レシーバ(BroadcastReceiver)

全アプリに対する ***公開要求種別応答要請*** への対応処理を担います。
応答要請に対して、無視や、任意のアクティビティを開始や、全アプリ共有の
告知領域(画面上端のステータスバー)やアラートを介して対応します。

コンテンツ プロバイダ(ContentProvider)

全アプリに対する ***独自データの標準取扱手順提供要請*** への対応処理を担います。

参考URL
android developers / フレームワークトピック > 開発の基礎 > アプリケーションのコンポーネント
http://developer.android.com/intl/ja/guide/topics/fundamentals.html



アプリ作成のまえに

Android アプリのコンポーネント

(コンポーネント・メソッド実装での注意事項)

Android システムから呼び出された、

コンポーネントの各種メソッド処理実装は、

数秒以内に終了できるものでなければなりません。

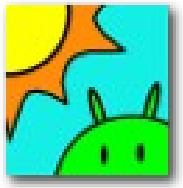
Android システムから呼び出される処理(コンポーネントの各種メソッド処理含む)は、
アプリケーション・プロセスのメイン・スレッド で実行されることになります。

- Android システムにメイン・スレッドを長時間返さないでいると ANR が発生します。
- 長時間かかる処理は、(上記の理由から)ワーカ・スレッドで実行させる必要があります。
- 画面表示の変更などは、メインス・レッドで実行される処理からでないと変更できません。
- Android システム呼出(メイン・スレッドでの実行)を任意の処理とタイミングで行わせるには、android.os.Handler や android.os.AsyncTask<Params, Progress, Result>を利用します。

参考URL

android developers / ベストプラクティス > Designing for Responsiveness
http://developer.android.com/intl/ja/guide/practices/design/responsiveness.html

ソフトウェア技術ドキュメントを勝手に翻訳 / ベストプラクティス > 5. レスポンスのための設計
http://sites.google.com/a/techdoctranslator.com/jp/android/practices/responsiveness



アプリ作成のまえに

Android アプリのコンポーネント

(アクティビティ と タスク および スタック)

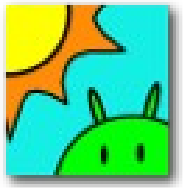
Android システムは、
一連のアプリ連携をユーザに提供するため、
アクティビティの呼出状態履歴を ***スタック(stack)*** と呼び、
特定アクティビティ関連の一塊のスタックを ***タスク(task)*** と呼びます。

一般的にアプリから別アプリのコンポーネントの要素を呼び出すことができますが、
ユーザ操作により画面遷移が随時変更可能かつ、一連の画面遷移(アプリ連携)が
ユーザ・エクスペリエンス(使い勝手の実感)のポイントとなるため、

画面遷移のコンポーネントであるアクティビティの呼出状態履歴のみ、タスクという
別枠の概念を追加して管理している？ようです。

タスクは、Intent のフラグと AndroidManifest.xml の <activity> 要素の属性設定の
相互作用により変更可能です。

参考URL
android developers / フレームワークトピック > 開発の基礎 > アクティビティとタスク
http://developer.android.com/intl/ja/guide/topics/fundamentals.html



アプリ作成のまえに

Android アプリのコンポーネント

(コンポーネントのライフサイクル)

Android アプリのコンポーネントには、

ライフタイム と *ライフサイクル(アプリ状態遷移)* があり、

ライフサイクル・メソッド に状態変化時の独自対応実装を行います。

Android アプリのコンポーネントは、他アプリのコンポーネント実行(マルチタスク実行とフォアグランド・アプリの変更)が任意に可能かつ、メモリ不足が発生すれば重要度の低いプロセスが削除されてしまうため、各アプリの実行状態は、不意に変化することとなります。

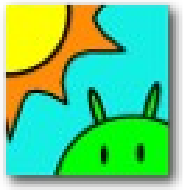
このためコンポーネントには、Android システムからアプリ状態の変化が通知されるライフサイクル・メソッドが予め用意され、各アプリは、そこに独自対応実装を行います。

各コンポーネントのライフサイクル・メソッド一覧

アクティビティ	void onCreate(Bundle savedInstanceState), void onStart(), void onRestart(), void onResume(), void onPause(), void onStop(), void onDestroy()
サービス	void onCreate(), void onStart(Intent intent), void onDestroy()
ブロードキャスト・レシーバ	void onReceive(Context curContext, Intent broadcastMsg)

参考URL

android developers / フレームワークトピック > 開発の基礎 > コンポーネントのライフサイクル
http://developer.android.com/intl/ja/guide/topics/fundamentals.html

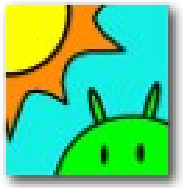


アプリ作成のまえに

Android (基礎)についてのおさらい

おさらいは、終了です

御協力、ありがとうございました。

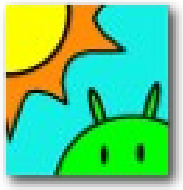


作成アプリについて

作成アプリの決定 (やりたいことからの着想)

- それなりに、実用性がないと作っても利用しないよね？
- Androidの基本的(特徴的)機能を盛り込みたいなぁ。
  (ネット接続や Intent を利用したアプリ連携、画面表示もちょっと変化をつけてみたいな)
- Android 標準だけど使っていないAPIを活用してみたいな。
  (XMLパーサは、これから色々応用できそう)
- 世の中クラウドばやりだし、WebAPIを利用したいな。
  (調べてみると、livedoor のお天気情報が簡易に利用できるみたい)

というわけで作成アプリは、
「今日の天気」に決定です



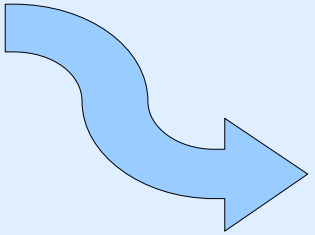
今日の天気アプリを
つくってみました

作成アプリについて

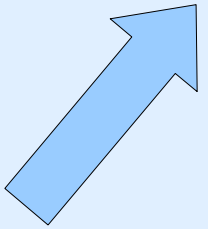
「今日の天気」アプリの機能

お天気Webサービス
Livedoor Weather Web Service

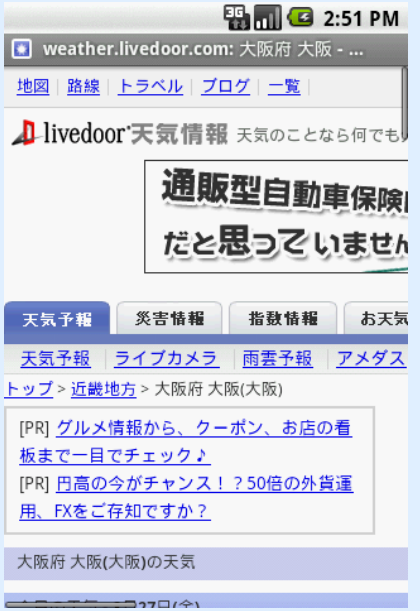
お天気情報を取得して、
解析した内容を
画面に表示



メイン画面



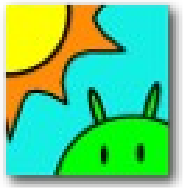
詳細情報が見たいときは、
libedoor天気情報を
ブラウザで確認



ブラウザ画面

地域情報を取得して、
解析した内容を
リストにして表示

地域選択画面



アプリ実装について

お天気情報の取得
(お天気Webサービス仕様)

お天気情報は、Livedoor Weather Web Service を利用します。

現在全国142カ所の今日・明日・あさっての天気予報・予想気温と
都道府県の天気概況情報が提供されています。

Q&Aより商用利用でなければ、自由に利用可能なそうです。

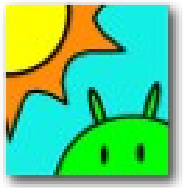
・お天気情報は、ベースとなるURLに地域などのパラメータを加えて
リクエストするだけで、XMLデータとして提供されます。

・リクエストに使用するパラメータ名とパラメータ値は、以下のとおり
city ：地域を表します。　　下記のid番号を値に利用します。
day ：予報日を指定します。今日の場合の値は、today です。

・リクエストする地域のパラメータ値は、全国の地点定義表（RSS）の
「1次細分区（cityタグ）」のidに対応しています。

ベースＵＲＬ	***http://weather.livedoor.com/forecast/webservice/rest/v1***
一次細分区定義表 - livedoor 天気情報	***http://weather.livedoor.com/forecast/rss/forecastmap.xml***

参考URL
http://weather.livedoor.com/weather_hacks/webservice.html　Livedoor Weather Hacks（お天気Webサービス仕様）



今日の天気アプリを
つくってみました

アプリ実装について

お天気情報の取得
(apache HttpClient の利用)

**サーバからの
XMLデータ取得には、
HttpClient を利用しています。**

http 接続によるデータ取得は、
java.net.HttpURLConnection を
利用しても可能ですが、

Basic 認証を行いたい場合は、
バグがあるそうなので、
HttpClientを利用しています。

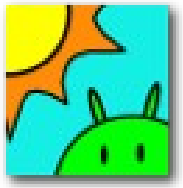
ちなみに、大阪の地域 id は、"81"です。

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import android.net.Uri;
import android.net.Uri.Builder;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

//URI作成 (お天気情報・取得データ指定)
UriBuilder uriBuilder = new Uri.Builder();
uriBuilder.scheme               ("http");
uriBuilder.authority            ("weather.livedoor.com");
uriBuilder.path                 ("/forecast/webservice/rest/v1");
uriBuilder.appendQueryParameter("city", cityId);
uriBuilder.appendQueryParameter("day",  "today");
/*
uriBuilder.fragment             ("SECTION1");
*/
String uri = uriBuilder.toString();
```

Http接続によるデータ取得サンプル (1/2)

参考URL

http://hc.apache.org/httpcomponents-client/index.html	Apache Software Foundation HttpClient
http://hc.apache.org/httpcomponents-client/tutorial/html/	HttpClient Tutorial
http://hc.apache.org/httpcomponents-client-4.0.1/examples.html	HttpClient Examples



今日の天気アプリを
つくってみました

アプリ実装について

お天気情報の取得
(apache HttpClient の利用)

**サーバからの
XMLデータ取得には、
HttpClient を利用しています。**

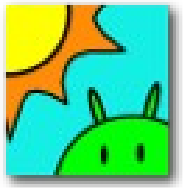
右記サンプルは、
HttpClient Examples の
Manual connection release を
参考にしています。

```
//Http接続によるデータ取得
HttpClient   httpclient = new DefaultHttpClient();
HttpGet      httpget    = new HttpGet(uri);
HttpResponse response   = httpclient.execute(httpget);
HttpEntity   entity     = response.getEntity();
//(※)HttpGet は、HttpUriRequest インターフェースを実装しています

if (entity != null) {
    BufferedReader reader = new BufferedReader(
                                new InputStreamReader(
                                  entity.getContent() ) );
    //(※)entity.getContent() は、InputStream を返します。
    try {
        System.out.println(reader.readLine());
    } catch (IOException ex) {
        throw ex;
    } catch (RuntimeException ex) {
        httpget.abort();
        throw ex;
    } finally {
        reader.close();
    }
}
```

http接続によるデータ取得サンプル (2/2)

参考URL

http://hc.apache.org/httpcomponents-client/index.html	Apache Software Foundation HttpClient
http://hc.apache.org/httpcomponents-client/tutorial/html/	HttpClient Tutorial
http://hc.apache.org/httpcomponents-client-4.0.1/examples.html	HttpClient Examples



アプリ実装について

お天気情報の解析
(XmlPullParser の利用)

**XMLデータ解析には、
XmlPullParser を
利用しています。**

XmlPullParser は、
SAX(Simple API for XML)のクラスです

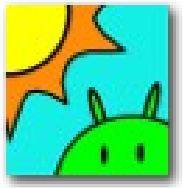
SAXはXMLを先頭から読み込んで
処理していくため、高速でメモリ消費も
少なくて済むそうですが、その反面
複雑な構造のXMLを読み込む場合に
制御が複雑になる弱点を持つそうです。

```
import android.util.Xml;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import java.io.IOException;
try{
   XmlPullParser parser = Xml.newPullParser();
   parser.setInput(new StringReader(
       "<tag1 attr1=¥"ATTR1¥" attr2=¥"ATTR2¥">TEXT1</tag1>"));
   int type;
   while((type = parser.next()) != XmlPullParser.END_DOCUMENT){
       if(type == XmlPullParser.START_TAG){
           System.out.println("TAG=" + parser.getName());
           for(int i = 0; i < parser.getAttributeCount(); i++){
               System.out.println("ATTRIBUTE=" +
                   parser.getAttributeName(i) + ":" +
                   parser.getAttributeValue(i));
           }
       }
       if(type == XmlPullParser.TEXT){
           System.out.println("TEXT="+parser.getText());
       }
   }
}catch(XmlPullParserException e){
}catch(IOException e){
}finally{
}
```

XmlPullParserによるXML解析サンプル

参考URL

http://www.xmlpull.org/v1/doc/api/org/xmlpull/v1/package-summary.html	xmlpull.org / Javadoc
http://www.adakoda.com/adakoda/2009/01/android-xmlpullparserxml.html	XmlPullParser(XML解析(パース))
http://d.hatena.ne.jp/d-kami/20100619/1276959474	XmlPullParserを使う
http://android.siprop.org/index.php?%CA%D9%B6%AF%B2%F1%2FXMLSQLite	XMLとSQLiteを使用する



アプリ実装について

地域情報のリスト
(ListViewとAdapter)

地域情報の画面には、ListViewが利用されています。

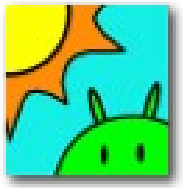
画面に追加してもらうために、SimpleAdapter に適用させる子要素(地域情報)のリストを生成します。

ListActivity を利用する場合、ListViewのリソース id は、*"@android:id/list"* 固定となるようです。

```
import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

import android.widget.ListView;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.SimpleAdapter;

//SimpleAdapter に適用させる、
//ListView に追加する子要素情報のリストを生成します。
//(※)サーバより取得した地域情報(リスト)から、
//    地域名 title と 地域識別番号 id というキー名をもつ
//    子要素のリストを生成しています。
final List<HashMap<String, String>> cityList =
    new ArrayList<HashMap<String, String>>();
for(CityData data : cityDataList){
    HashMap<String, String> map =
        new HashMap<String, String>();
    map.put("title", data.title.value);
    map.put("id",    data.id.value);
    cityList.add(map);
}
```

SimpleAdapterによる子要素追加サンプル (1/2)

参考URL
android developers / チュートリアル > Hello, Views > List View
http://developer.android.com/intl/ja/resources/tutorials/views/hello-listview.html
ソフトウェア技術ドキュメントを勝手に翻訳 / チュートリアル? > 2. Hello Views > 2.6 リストビュー
http://sites.google.com/a/techdoctranslator.com/jp/resources/tutorials/views/hello-listview
http://d.hatena.ne.jp/isher/20091009/1255029532 isherの日記 / 2009-10-09 ListViewについて



アプリ実装について

地域情報のリスト
(ListViewとAdapter)

**地域情報の画面には、
ListViewが利用されています。**

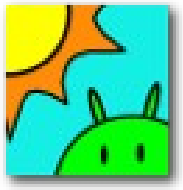
ListView の Adapter に、
子要素(地域情報)のリストが
設定された SimpleAdapter を
指定することにより、

画面に地域情報のリストを
追加してもらっています。

```
//SimpleAdapter オブジェクトに、
//子要素情報のリストと子要素構成の情報を設定します。
//(※)子要素構成は、title と id というキー名を持つことと、
//    子要素構成のレイアウト情報の参照先(*1)および、
//    子要素の title と id のリソース情報を指定しています。
//    (*1)layout リソースの city_info.xml
SimpleAdapter adapter = new SimpleAdapter(
        this,
        cityList,
        R.layout.city_info,
        new String[]{"title", "id"},
        new int[]{R.id.title, R.id.id});

//ListView の Adapter に、
//SimpleAdapter オブジェクトを設定します。
//(※)Adapter は、View に追加する子要素の適用指定を担います。
ListView list = (ListView)findViewById(android.R.id.list);
list.setAdapter(adapter);
```

SimpleAdapterによる子要素追加サンプル (2/2)

参考URL
android developers / チュートリアル > Hello, Views > List View
http://developer.android.com/intl/ja/resources/tutorials/views/hello-listview.html
ソフトウェア技術ドキュメントを勝手に翻訳 / チュートリアル? > 2. Hello Views > 2.6 リストビュー
http://sites.google.com/a/techdoctranslator.com/jp/resources/tutorials/views/hello-listview
http://d.hatena.ne.jp/isher/20091009/1255029532 isherの日記 / 2009-10-09 ListViewについて

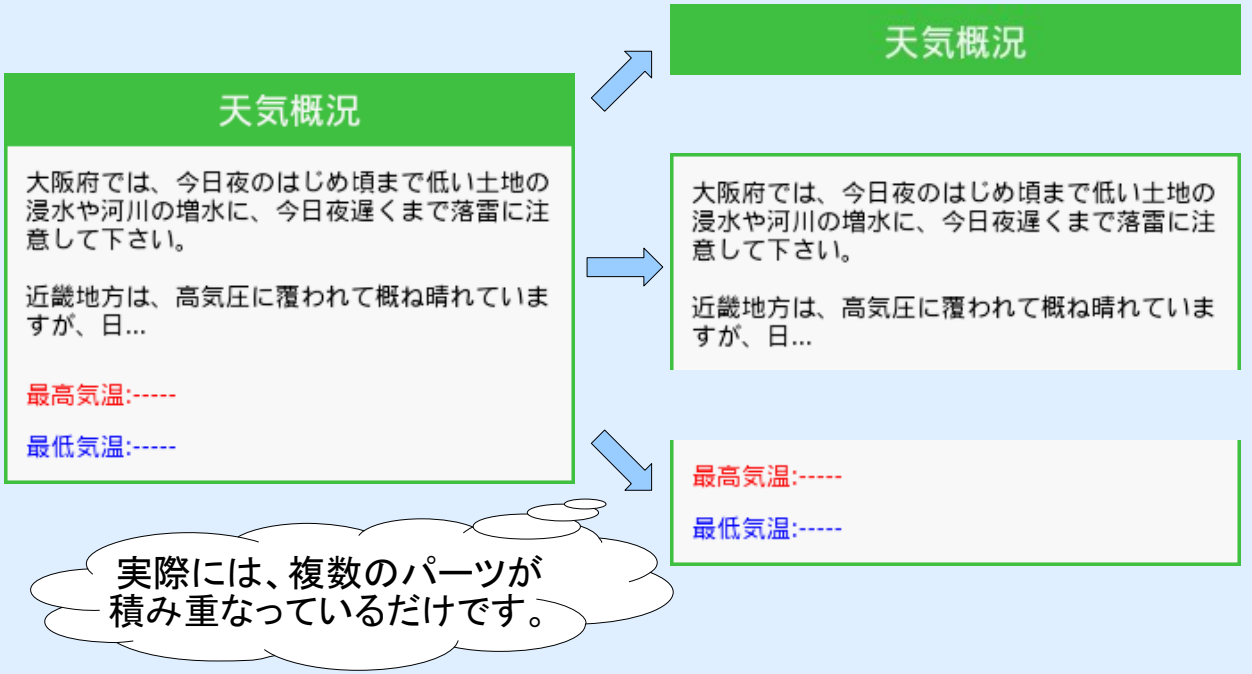


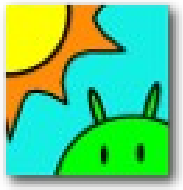
アプリ実装について

天気概要の枠表示

メイン画面は、2重の LinearLayout の入れ子の中心に
Widget を配した矩形を積み重ねて作成しています。

天気概要の緑色の枠は、
外側の LinearLayout に緑色で 2pixcel のマージンを持たせ、
境界がないようにする辺(例えば、概況文と気温の接合辺)は、
背景と同じ色に設定することで表現しました。





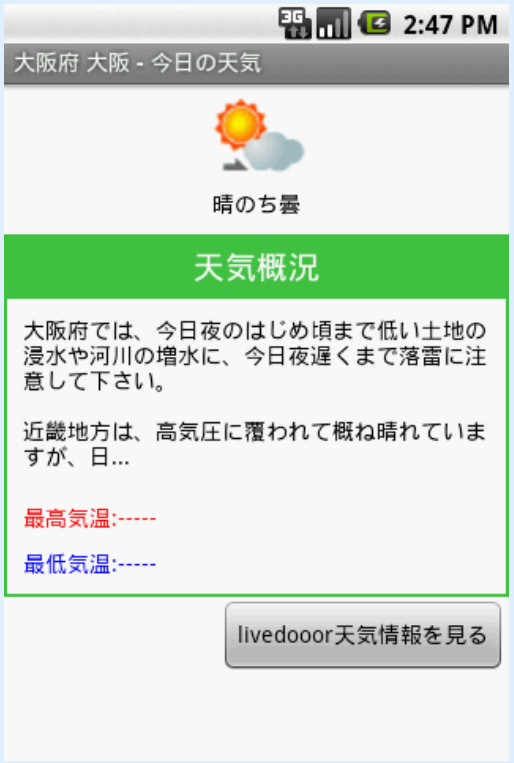
今日の天気アプリを
つくってみました

アプリ実装について

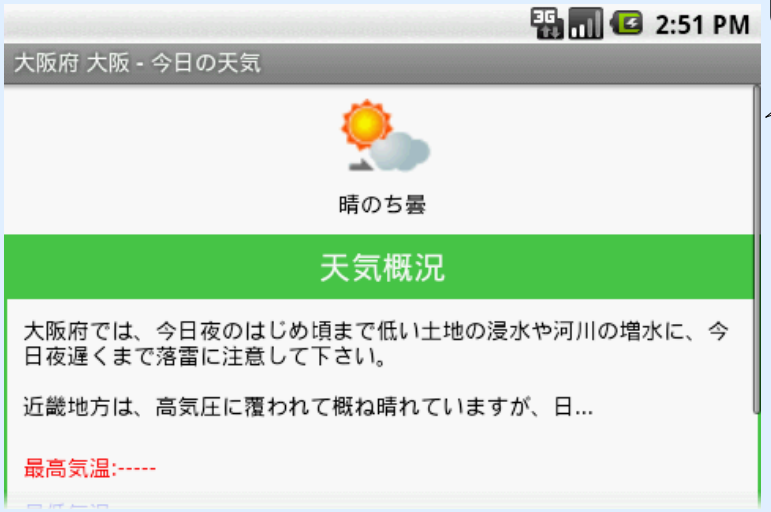
その他

**メイン画面のルートViewは、
ScrollView に指定されています。**

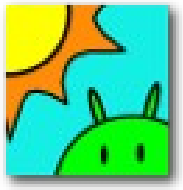
**この指定により、1画面に表示しきれない場合でも、
自動的にスクロールバーが表示されて、
スクロール閲覧可能になっています。**



Landscapeへ



Landscape 表示切替で一度に表示しきれなくなる場合でも、スクロール閲覧ができます。



今日の天気アプリを
つくってみました

作成アプリについて

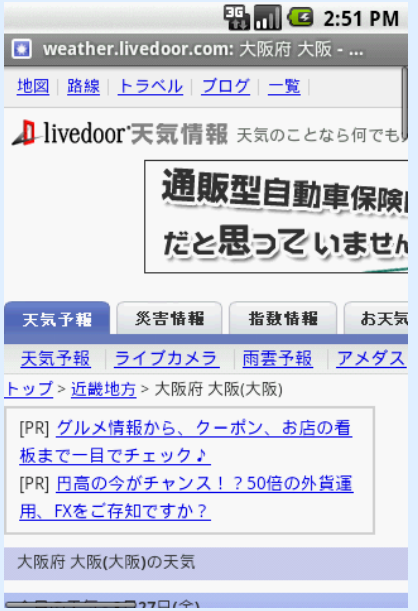
アプリで利用した Android の機能

お天気Webサービス
Livedoor Weather Web Service

HttpClient による接続と、
XMLデータ取得

XmlPullParserによる
XML内容解析

メイン画面



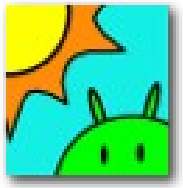
ブラウザ画面

Intent に Action と data を指定した、
データ指定つきのアプリケーション連携

地域選択画面

各地の地域情報を
Adapter を利用して
ListView のリストに
追加

ライフサイクル・メソッドへの実装、
サーバ上からの画像データ取得と、
ImageView による天気画像表示、
Layout の入れ子による枠表示や、
Activity でのMenuとDialogの作成

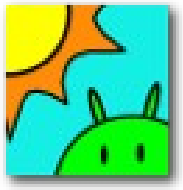


作成アプリについて

今回、積み残したもの

- **ワーカスレッドによる、サーバからのデータ取得処理対応**
  メイン・スレッドでデータ取得を行っているので、データ取得が長時間となる場合に
  ANRが発生する恐れがあります。 また AsyncTask などで、データ取得後の処理を
  メイン・スレッドから呼び出してもらうようにする対応も必要でしょう。
- **設定データのプリファレンスへの保管対応**
  ユーザにより選択された予報地域は、プリファレンスに保管するようにして、
  再起動時も以前の設定が有効になるようにする必要があるでしょう。
- **見栄えや使い心地を良くするデザインの改良**
  現時点の画面デザインは、基本テクニックのみの最低限レベルにとどまっています。

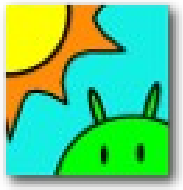
**その他にも、未発覚のバグ対応など、
対応が必要なことは、山ほどあるでしょう
時間がないよう... orz**



今日の天気アプリを
つくってみました

ご静聴、

ありがとうございました。



参考資料

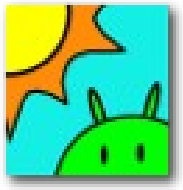
「今日の天気」アプリのソース＆リソース

以降より、「今日の天気」アプリの

全てのソースとリソースを公開します。

適当に作成したアプリのソースですから、

お勧めできる実装でないので、参考程度に見てくださいね。



Manifest & XML Resource

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.cch_lab.android.sample.tinyweatherinfo"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/otenki_icon"
                 android:label="今日の天気">
        <activity android:name=".WeatherInfoActivity"
                  android:label="今日の天気">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".CityListActivity"
                  android:label="地域情報">
        </activity>

    </application>

    <uses-sdk android:minSdkVersion="4" />
    <uses-permission android:name="android.permission.INTERNET">
    </uses-permission>

</manifest>
```

label 属性に直接文字指定していますが、

一般的に文字リソースは、
android:label="@string/app_name"
のように記述してから、

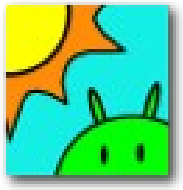
res/value/strings.xml のXMLデータに、
実態を記述した方が良いと思います。

res/layout/city_info.xml

```
<?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="wrap_content"
    >
<TextView
    android:id="@+id/title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    android:textSize="20sp"
    />
<TextView
    android:id="@+id/id"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_margin="2dp"
    android:textSize="15sp"
    />
</LinearLayout>
```

res/layout/city_list.xml

```
<?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"
    >
<ListView
    android:id="@android:id/list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:cacheColorHint="#000000"
    />
</LinearLayout>
```



今日の天気アプリを
つくってみました

Image Resource

res/drawable-hdpi/otenki_icon.png

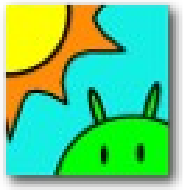
72×72

res/drawable-mdpi/otenki_icon.png

48×48

res/drawable-ldpi/otenki_icon.png

36×36



今日の天気アプリを
つくってみました

src/com/cch_lab/android/sample/tinyweatherinfo/ WeatherInfoActivity.java (1/5)

```
package com.cch_lab.android.sample.tinyweatherinfo;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.Window;
import android.view.View;
import android.view.Gravity;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Button;
import android.widget.ScrollView;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.widget.TableLayout.LayoutParams;
import android.graphics.drawable.Drawable;
import android.graphics.drawable.BitmapDrawable;

import android.net.Uri;
import android.util.Xml;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;

import java.util.ArrayList;

import org.json.JSONObject;
import org.json.JSONArray;
import org.json.JSONException;

import com.cch_lab.android.sample.tinyweatherinfo.XMLUtil;
import com.cch_lab.android.sample.tinyweatherinfo.XMLUtil.KeyInfo;
import com.cch_lab.android.sample.tinyweatherinfo.XMLUtil.KeyValue;

import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;

public class WeatherInfoActivity extends Activity {
    private final Object        INSTANCE     = this;
    private XMLUtil             weatherXML   = null;
    private static String       city_id      = "81";  //天気情報を取得する
                                                      //地域のID番号
                                                      //(初期値は、大阪市)
    private static final int    CITY_ID_CODE = 100;   //地域情報のコード
    public  static final String CITY_ID_KEY  = "city_id"; //地域情報のキー
```

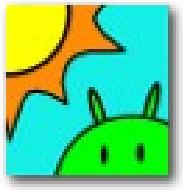
ソースを見て、
Viewオブジェクトの入れ子構造が判るよう、オブジェクトを直接操作していますが、

一般的にレイアウトは、
layout リソースにXMLデータで記述した方が良いと思います。

「今日の天気」アプリ
メイン画面のアクティビティ

```
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
}

@Override
protected void onResume() {
    super.onResume();

    //画面  View ルート・オブジェクトの設定
    ScrollView scrollView = new ScrollView(this);
    scrollView.setBackgroundColor(Color.rgb(255,255,255));
    setContentView(scrollView);
    //
    LinearLayout layout = new LinearLayout(this);
    layout.setOrientation(LinearLayout.VERTICAL);
    scrollView.addView(layout);

    //天気情報の取得
    final WeatherData data  = getWeatherData();
    if(data != null){
    //    Toast.makeText(this, "天気予報情報を取得しました",
    //        Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(this, "天気予報情報が取得できませんでした",
            Toast.LENGTH_LONG).show();
    }

    try{
        if(data != null){
            //タイトル設定
            setTitle(data.title.value);
            setTitleColor(Color.rgb(255,255,255));

            //天気画像
            ImageView weatherImage = new ImageView(this);
            weatherImage.setImageDrawable(
                getImage(data.image_url.value));
            weatherImage.setMinimumWidth(
                Integer.parseInt(data.image_width.value)*2);
            weatherImage.setMinimumHeight(
                Integer.parseInt(data.image_height.value)*2);
            weatherImage.setPadding(10,10,10,5);
            setLayoutParametor(weatherImage,
              LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
            //
            LinearLayout imageFrame = new LinearLayout(this);
            imageFrame.setBackgroundColor(Color.rgb(255,255,255));
            imageFrame.setGravity(Gravity.CENTER);
            imageFrame.addView(weatherImage);
            //
            layout.addView(imageFrame);
```

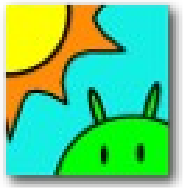
ルート View に
ScrollViewを指定しています



src/com/cch_lab/android/sample/tinyweatherinfo/ WeatherInfoActivity.java (2/5)

```
//天気キャプション
TextView weatherCaption = new TextView(this);
weatherCaption.setText(data.telop.value);
weatherCaption.setTextColor(Color.rgb(0,0,0));
weatherCaption.setPadding(10,5,10,10);
setLayoutParametor(weatherCaption,
  LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
//
LinearLayout captionFrame = new LinearLayout(this);
captionFrame.setBackgroundColor(Color.rgb(255,255,255));
captionFrame.setGravity(Gravity.CENTER);
captionFrame.addView(weatherCaption);
//
layout.addView(captionFrame);


//天気概況タイトル
TextView descTaitle = new TextView(this);
descTaitle.setText("天気概況");
descTaitle.setTextSize(20.0f);
descTaitle.setTextColor(Color.rgb(255,255,255));
descTaitle.setPadding(5,5,5,5);
descTaitle.setBackgroundColor(Color.rgb(64,192,64));
setLayoutParametor(descTaitle,
  LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
//
LinearLayout titleFrame = new LinearLayout(this);
titleFrame.setBackgroundColor(Color.rgb(64,192,64));
titleFrame.setPadding(2,2,2,2);
titleFrame.setGravity(Gravity.CENTER);
titleFrame.addView(descTaitle);
//
layout.addView(titleFrame);


//天気概況
TextView weatherDesc = new TextView(this);
weatherDesc.setTextColor(Color.rgb(0,0,0));
weatherDesc.setText(data.description.value);
weatherDesc.setPadding(10,10,10,10);
weatherDesc.setBackgroundColor(Color.rgb(255,255,255));
setLayoutParametor(weatherDesc,
  LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT);
//
LinearLayout descFrame = new LinearLayout(this);
descFrame.setBackgroundColor(Color.rgb(64,192,64));
descFrame.setPadding(2,0,2,0);
descFrame.setGravity(Gravity.LEFT);
descFrame.addView(weatherDesc);
//
layout.addView(descFrame);
```

```
//最高気温
String maxTemp = data.max_temperature.value;
if(maxTemp == null || "".equals(maxTemp)){
    maxTemp = "-----";
} else {
    maxTemp = maxTemp + "℃";
}
TextView maxTemper = new TextView(this);
maxTemper.setTextColor(Color.rgb(255,0,0));
maxTemper.setText("最高気温:" + maxTemp);
maxTemper.setPadding(10,10,10,5);
maxTemper.setBackgroundColor(Color.rgb(255,255,255));
setLayoutParametor(maxTemper,
  LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT);
//
LinearLayout maxTempFrame = new LinearLayout(this);
maxTempFrame.setBackgroundColor(Color.rgb(64,192,64));
maxTempFrame.setPadding(2,0,2,0);
maxTempFrame.setGravity(Gravity.LEFT);
maxTempFrame.addView(maxTemper);
//
layout.addView(maxTempFrame);


//最低気温
String minTemp = data.min_temperature.value;
if(minTemp == null || "".equals(minTemp)){
    minTemp = "-----";
} else {
    minTemp = minTemp + "℃";
}
TextView minTemper = new TextView(this);
minTemper.setTextColor(Color.rgb(0,0,255));
minTemper.setText("最低気温:" + minTemp);
minTemper.setPadding(10,5,10,10);
minTemper.setBackgroundColor(Color.rgb(255,255,255));
setLayoutParametor(minTemper,
  LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT);
//
LinearLayout minTempFrame = new LinearLayout(this);
minTempFrame.setBackgroundColor(Color.rgb(64,192,64));
minTempFrame.setPadding(2,0,2,2);
minTempFrame.setGravity(Gravity.LEFT);
minTempFrame.addView(minTemper);
//
layout.addView(minTempFrame);
```



今日の天気アプリを
つくってみました

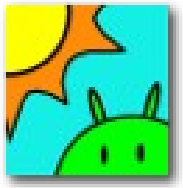
src/com/cch_lab/android/sample/tinyweatherinfo/WeatherInfoActivity.java (3/5)

```
            //リンク・ボタン生成
            Button button = new Button(this);
            button.setText("livedooor天気情報を見る");
            button.setOnClickListener(
                    new View.OnClickListener(){
                        public void onClick(View view){
                            Intent intent = new Intent(
                                "android.intent.action.VIEW",
                                Uri.parse(data.link.value));
                            WeatherInfoActivity.this.
                                startActivity(intent);
                        }
                    }
                );
            setLayoutParametor(button,
              LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT);
            //
            LinearLayout buttonFrame = new LinearLayout(this);
            buttonFrame.setBackgroundColor(Color.rgb(255,255,255));
            buttonFrame.setPadding(2,2,2,2);
            buttonFrame.setGravity(Gravity.RIGHT);
            buttonFrame.addView(button);
            //
            layout.addView(buttonFrame);
        }
    }
    catch(Exception e){
        Toast.makeText(this, "天気予報情報画面の作成に失敗しました",
            Toast.LENGTH_LONG).show();
        Util.log(INSTANCE, "Error " + e.getClass().getSimpleName()
            + "occer." + " message =¥""+e.getMessage()+"¥"");
    }
    finally{
        ;
    }
}

@Override
protected void onActivityResult(
    int requestCode, int resultCode, Intent data) {

    //地域情報画面からの呼出結果を取得する
    if(requestCode == CITY_ID_CODE
        && resultCode == Activity.RESULT_OK){
        if(data.getExtras() != null){
            city_id = data.getExtras().getString(CITY_ID_KEY);
            //Toast.makeText(getApplicationContext(),
            //   "city_id="+city_id, Toast.LENGTH_SHORT).show();
        }
    }
}
```

Intent の Action に View を、データにlibedoor天気情報のhtmlサーバアドレスを指定して、データを指定したアプリ起動を行っています。

```
/**
 * View オブジェクトへのレイアウト・パラメータ設定.
 * <p>View オブジェクトにレイアウト・パラメータを設定します。</p>
 * レイアウト幅とレイアウト高は、
 * android.view.ViewGroup.LayoutParams の固定値
 * {FILL_PARENT(またはMATCH_PARENT)か WRAP_CONTENT}を
 * 指定してください
 *
 * @param view          レイアウトパラメータを設定する、
 *                      View オブジェクト
 * @param layout_width  レイアウト幅
 * @param layout_height レイアウト高
 */
public void setLayoutParametor(
    View view, int layout_width, int layout_height){
    view.setLayoutParams(
        new ViewGroup.LayoutParams(layout_width, layout_height));
}

private final int MENU_GROUP_ID_FIRST    = 1;
private final int MENU_ITEM_ID_CITY_LIST = 1;
private final int MENU_ITEM_ID_ABOUT     = 2;
private final int DIALOG_ID_ABOUT        = 2;

@Override
public boolean onCreateOptionsMenu(Menu menu){
    super.onCreateOptionsMenu(menu);
    //Menu#add(int groupId, int itemId, int order, int titleRes)
    MenuItem menuItem1 = menu.add(
            MENU_GROUP_ID_FIRST,     //メニュー・グループのID
                                     //※NONEは、グループを指定しない
            MENU_ITEM_ID_CITY_LIST, //メニュー・アイテムのID
                                     //※NONEは、アイテムを識別しない
            Menu.NONE,               //メニュー・アイテムのソート順種別
                                     //※NONEは、ソート順を指定しない
            "地域指定");             //メニュー・アイテムの項目名

    MenuItem menuItem2 = menu.add(
            MENU_GROUP_ID_FIRST,    //メニュー・グループのID
                                    //※NONEは、グループを指定しない
            MENU_ITEM_ID_ABOUT,     //メニュー・アイテムのID
                                    //※NONEは、アイテムを識別しない
            Menu.NONE,              //メニュー・アイテムのソート順種別
                                    //※NONEは、ソート順を指定しない
            "アプリについて");      //メニュー・アイテムの項目名
    return true;
}
```

Activity メニューの作成



今日の天気アプリを
つくってみました

src/com/cch_lab/android/sample/tinyweatherinfo/ WeatherInfoActivity.java (4/5)

Activity メニューの
項目選択処理

```
@Override
public boolean onOptionsItemSelected(MenuItem menuItem){
    super.onOptionsItemSelected(menuItem);
    switch(menuItem.getItemId()){
        case MENU_ITEM_ID_CITY_LIST :
            //地域情報画面に遷移させます (結果を返却させる画面呼出)
            Intent intent = new Intent(
                WeatherInfoActivity.this, CityListActivity.class);
            WeatherInfoActivity.this
                .startActivityForResult(intent, CITY_ID_CODE);
            break;
        case MENU_ITEM_ID_ABOUT :
            showDialog(DIALOG_ID_ABOUT);
            break;
        default:
            break;
    }
    return true;
}
```

Activity ダイアログの作成

```
@Override
protected Dialog onCreateDialog(int id){
    Dialog dialog = null;
    switch(id){
        case DIALOG_ID_ABOUT :
            dialog = new AlertDialog.Builder(this)
                .setTitle("アプリについて")
                .setIcon(R.drawable.otenki_icon)
                .setMessage(
"天気情報は、livedoor様の¥n Weather Hacks を利用させてもらっています。")
                .create();
            break;
        default:
            break;
    }
    return dialog;
}
```

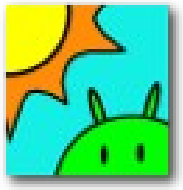
天気情報ウェブサービスから
天気情報のXMLデータ取得と、
取得XMLデータの解析処理

```
public Drawable getImage(String url) {
    HttpUtil http     = new HttpUtil();
    Drawable drawable = null;
    try{
        http.openHttpConnection(url);
        drawable = new BitmapDrawable(http.openHttpInputStream())
                .mutate();
    }
    catch(Exception e){
        Util.log(INSTANCE, "Image get failed.");
    }
    finally{
        http.closeHttpInputStream();
        http.closeHttpConnection();
    }
    return drawable;
}
```

パラメータをURIに直接記述しないで、
android.net.Uri.Builder を利用して、
URI を生成したほうが良いでしょう。

```
public WeatherData getWeatherData() {
    WeatherData data = null;
    HttpUtil    http = new HttpUtil();
    String      FORECUST_WEATHER_URI = //Livedoor お天気Webサービス
"http://weather.livedoor.com/forecast/webservice/rest/v1?city=" + city_id
+ "&day=today";

    try{
        if(weatherXML == null){
            weatherXML = new XMLUtil();
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
1,  "タイトル",      new String[]{"lwws","title"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
2,  "リンク",        new String[]{"lwws","link"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
3,  "予報発表時刻",  new String[]{"lwws","publictime"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
4,  "天気",          new String[]{"lwws","telop"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
5,  "天気概況",      new String[]{"lwws","description"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
6,  "天気画像URL",   new String[]{"lwws","image","url"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
7,  "天気画像幅",    new String[]{"lwws","image","width"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
8,  "天気画像高",    new String[]{"lwws","image","height"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
9,  "最高気温",      new String[]{"lwws","temperature","max","celsius"}));
weatherXML.addKeyInfo(XMLUtil.createKeyInfo(
10, "最低気温",      new String[]{"lwws","temperature","min","celsius"}));
        }
```

天気情報のXMLユーティリティ・オブジェクトを生成し、
天気情報用のXMLデータ構造を設定するために、
特定ID番号とキー名に対応させるタグ構成を指定した、
キー情報(KeyInfo)を順次追加しています。



src/com/cch_lab/android/sample/tinyweatherinfo/ WeatherInfoActivity.java (5/5)

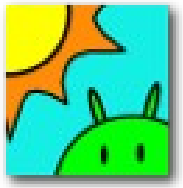
```
            XmlPullParser parser = Xml.newPullParser();
            http.openHttpConnection(FORECUST_WEATHER_URI);
            parser.setInput(http.openHttpInputStreamReader());
            weatherXML.setParser(parser);
            while(!weatherXML.parse()){
                for(XMLUtil.KeyValue keyValue : weatherXML.getKeyValues()){
                    if(data == null) data = new WeatherData();
                    int keyId = keyValue.key.ID_NUM;
                    switch(keyId){
case  1: data.title           = keyValue; break;
case  2: data.link            = keyValue; break;
case  3: data.publictime      = keyValue; break;
case  4: data.telop           = keyValue; break;
case  5: String desc = keyValue.value.replaceFirst("¥¥.¥¥.¥¥..*$", "...");
         data.description = XMLUtil.createKeyValue(keyValue.key, desc);
         break;
case  6: data.image_url       = keyValue; break;
case  7: data.image_width     = keyValue; break;
case  8: data.image_height    = keyValue; break;
case  9: data.max_temperature = keyValue; break;
case 10: data.min_temperature = keyValue; break;
default: break;
                    }
                }
            }
            Util.log(INSTANCE, "XML get success.");
        }
        catch(Exception e){
            Util.log(INSTANCE, "XML get failed. (" + e.getMessage() +")");
        }
        finally{
            http.closeHttpInputStreamReader();
            http.closeHttpConnection();
        }
        return data;
    }
```

```
class WeatherData {
    private KeyValue init = XMLUtil.createKeyValue(
                                XMLUtil.createKeyInfo(0, "", null),
                                ""); //初期値
    public  KeyValue title           = init; //大阪府 大阪-今日の天気
    public  KeyValue link            = init; //livedoor 天気情報のURL
    public  KeyValue publictime      = init; //予報発表時刻
    public  KeyValue telop           = init; //天気
    public  KeyValue description     = init; //天気概況文
    public  KeyValue image_url       = init; //天気画像 URL
    public  KeyValue image_width     = init; //天気画像 幅
    public  KeyValue image_height    = init; //天気画像 高
    public  KeyValue max_temperature = init; //最高気温 (celsius)
    public  KeyValue min_temperature = init; //最低気温 (celsius)
```

天気情報データ・保管用クラス

天気情報用のXMLデータ構造が設定済みなので、
XMLユーティリティ・オブジェクトのparseメソッドを用いれば、
解析タグ構成が、設定済のキー情報(KeyInfo)に合致して
いた場合、そのKeyValueオブジェクト(のリスト)が取得できます。

KeyValueが取得できれば、ID番号も取得できるので、
switch 文で、各特定ID番号(キー)用の対応を行っています。



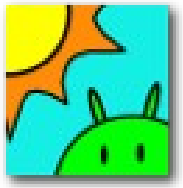
今日の天気アプリを
つくってみました

src/com/cch_lab/android/sample/tinyweatherinfo/ CityListActivity.java (1/2)

SimpleAdapter を利用して、
ListView に取得地域情報の
リストを追加させます。

「今日の天気」アプリ
地域選択画面のアクティビティ

```
package com.cch_lab.android.sample.tinyweatherinfo;

import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.app.ListActivity;
import android.widget.ListView;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.SimpleAdapter;

import android.widget.Toast;

import android.util.Xml;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;

import java.util.List;
import java.util.ArrayList;
import java.util.HashMap;

import com.cch_lab.android.sample.tinyweatherinfo.XMLUtil;
import com.cch_lab.android.sample.tinyweatherinfo.XMLUtil.KeyInfo;
import com.cch_lab.android.sample.tinyweatherinfo.XMLUtil.KeyValue;


public class CityListActivity extends ListActivity {
    private final Object INSTANCE = this;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //画面  View ルート・オブジェクトの設定
        setContentView(R.layout.city_list);

        //地域情報の取得
        final ArrayList<CityData> cityDataList = getCityData();
        if(cityDataList.size() != 0){
        //  Toast.makeText(this, "地域情報を取得しました",
        //  Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(this, "地域情報が取得できませんでした",
            Toast.LENGTH_LONG).show();
            finish();
            Return;
        }

        //SimpleAdapter に適用させる、
        //ListView に追加する子要素情報のリストを生成
        //(※)取得した地域情報から、
        //    地域名 title と 地域識別番号 id というキー名をもつ
        //    子要素のリストを生成しています。
        final List<HashMap<String, String>>
        cityList = new ArrayList<HashMap<String, String>>();
        for(CityData data : cityDataList){
            HashMap<String, String> map = new HashMap<String,String>();
            map.put("title", data.title.value);
            map.put("id",    data.id.value);
            cityList.add(map);
        }

        //SimpleAdapter オブジェクトに、
        //子要素情報のリストと子要素構成の情報を設定
        //(※)子要素構成は、title と id というキー名を持つことと、
        //    子要素構成のレイアウト情報(layoutリソースのcity_info.xml)
        //    の参照先および、
        //    子要素の title と id のリソース情報を指定しています。、
        SimpleAdapter adapter = new SimpleAdapter(this,
                cityList, R.layout.city_info,
                new String[]{"title", "id"},
                new int[]{R.id.title, R.id.id});

        //ListView の Adapter に 、SimpleAdapter オブジェクトを設定
        ListView list = (ListView)findViewById(android.R.id.list);
        list.setAdapter(adapter);

        //ListView の子要素選択時のイベントを設定
        list.setOnItemClickListener(new OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View view,
                                    int position, long id) {
                //ListView でクリックされた子要素の位置を示す、
                //引数 position の数値は、
                //SimpleAdapter で指定したリスト(cityList)の
                //要素番号に相当するので、
                //cityList から、どの地域の id が選択されたのかを
                //取得しています。
                String city_id = cityList.get(position).get("id");

                if(city_id != null && !"".equals(city_id)){
                    //取得結果を呼出元に返却します。
                    Intent resultIntent = new Intent();
                    resultIntent.putExtra(
                        WeatherInfoActivity.CITY_ID_KEY, city_id);
                    setResult(Activity.RESULT_OK, resultIntent);
                    finish();
                }
            }
        });
    }
```



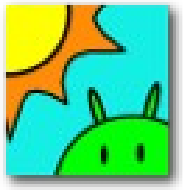
src/com/cch_lab/android/sample/tinyweatherinfo/ CityListActivity.java (2/2)

```
    public ArrayList<CityData> getCityData() {
        ArrayList<CityData> citydataList = new ArrayList<CityData>();
        String FORECUST_MAP_URI = //一次細分区定義表-livedoor 天気情報
        "http://weather.livedoor.com/forecast/rss/forecastmap.xml";
        HttpUtil http = new HttpUtil();
        try{
            XMLUtil cityXML = null;
            if(cityXML == null){
                cityXML = new XMLUtil();
cityXML.addKeyInfo(XMLUtil.createKeyInfo(1,  "地方",
new String[]{"rss","channel","source","area"}, "title"));
cityXML.addKeyInfo(XMLUtil.createKeyInfo(2,  "都道府県",
new String[]{"rss","channel","source","area","pref"},"title"));
cityXML.addKeyInfo(XMLUtil.createKeyInfo(3,  "市町村",
new String[]{"rss","channel","source","area","pref","city"}, "title"));
cityXML.addKeyInfo(XMLUtil.createKeyInfo(4,  "市町村 ID",
new String[]{"rss","channel","source","area","pref","city"}, "id"));
            }

            XmlPullParser parser = Xml.newPullParser();
            http.openHttpConnection(FORECUST_MAP_URI);
            parser.setInput(http.openHttpInputStreamReader());
            cityXML.setParser(parser);
            while(!cityXML.parse()){
                CityData data = new CityData();
                for(XMLUtil.KeyValue keyValue: cityXML.getKeyValues()){
                    int keyId = keyValue.key.ID_NUM;
                    switch(keyId){
    case  1: data.title = keyValue; citydataList.add(data); break;
    case  2: data.title = keyValue; citydataList.add(data); break;
    case  3: data.title = keyValue; citydataList.add(data); break;
    case  4: data.id    = keyValue; break;
    default: break;
                    }
                }
            }
            Util.log(INSTANCE, "XML get success.");
        }
        catch(Exception e){
            Util.log(INSTANCE, "XML MAP get failed. (" +
            e.getMessage() +")");
        }
        finally{
            http.closeHttpInputStreamReader();
            http.closeHttpConnection();
        }
        return citydataList;
    }
```

地域情報ウェブサービスから
地域情報のXMLデータ取得と、
取得XMLデータの解析処理

```
    class CityData {
        private KeyValue init = XMLUtil.createKeyValue(
                                XMLUtil.createKeyInfo(0, "", null, ""),
                                    ""); //初期値
        public  KeyValue title = init;  //地域名称
        public  KeyValue id    = init;  //地域 ID 番号
                                        //(※)地域区分が市町村時のみ
    }
}
```

地域情報データ・保管用クラス



今日の天気アプリを
つくってみました

src/com/cch_lab/android/sample/tinyweatherinfo/ XMLUtil.java (1/4)

```
package com.cch_lab.android.sample.tinyweatherinfo;

import java.io.IOException;
import android.util.Xml;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import java.util.ArrayList;

public class XMLUtil {
    private final Object INSTANCE = this;
    private static final XMLUtil DEFAULT = new XMLUtil();
    private XmlPullParser        parser;
    private ArrayList<KeyValue>  currKeyValues;
    private final int            MAX_DEPTH = 20;
    private String[]             tagName   = new String[MAX_DEPTH];
    private ArrayList<KeyInfo>   keyInfoList;

    public XMLUtil(){
        this.keyInfoList = new ArrayList<KeyInfo>();
    }

    public static KeyInfo createKeyInfo(
        int id, String desc, String[] tag){
        return DEFAULT.new KeyInfo(id, desc, tag);
    }

    public static KeyInfo createKeyInfo(
        int id, String desc, String[] tag, String attr){
        return DEFAULT.new KeyInfo(id, desc, tag, attr);
    }

    public static KeyValue createKeyValue(
        KeyInfo keyInfo, String value){
        return DEFAULT.new KeyValue(keyInfo, value);
    }

    public void addKeyInfo(KeyInfo keyInfo){
        keyInfoList.add(keyInfo);
    }

    public void setParser(XmlPullParser parser) {
        this.parser = parser;
    }
```

```
public boolean parse()throws IOException, XmlPullParserException {
    //(※)XmlPullParser#next() メソッドは、
    //    １項目を解析するたびに、
    //   当該項目の親項目からの全構造のタイプの出力を繰り返します。
    //   例：<tag1><tag2>text</tag2></tag1>のテキスト解析時は、
    //      [tag1],[tag1,tag2],[tag1,tag2,text]...というような
    //      タイプ出力となります。
    int     type  = parser.next();
    boolean isEnd = false;
    currKeyValues = new ArrayList<KeyValue>();
    switch(type){
        case XmlPullParser.START_DOCUMENT:
            break;
        case XmlPullParser.START_TAG :
            tagName[parser.getDepth()-1] = parser.getName();
            currKeyValues = getKeyValues(false);
            break;
        case XmlPullParser.TEXT:
            currKeyValues = getKeyValues(true);
            break;
        case XmlPullParser.END_TAG:
            break;
        case XmlPullParser.END_DOCUMENT:
            isEnd = true;
            break;
        default:
            break;
    }
    return isEnd;
}

public ArrayList<KeyValue> getKeyValues(){
    return currKeyValues;
}
```

当該XML解析用ユーティリティクラスは、
XmlPullParser が、XMLデータを走査して
要素や属性を返すだけの仕組みであるため、

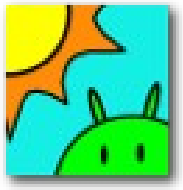
指定された要素構成に合致する場合のみ
簡易に対処できるよう、ID番号を伴った
テキストあるいは、指定属性名の内容を
Key, Value 形式で取得できるようにして、

XMLの要素構成ごとに、独自実装を記述
する手間を軽減する目的のクラスです。

src/com/cch_lab/android/sample/tinyweatherinfo/ XMLUtil.java (2/4)

```
/**
 * キー値取得.
 * 現在解析中のタグ構成(属性名含)が、
 * 設定済みのキー情報と合致している場合、
 * そのキー情報と値(のリスト)を返します。
 * @param  isText テキスト値取得フラグ
 *         (※)テキスト値を取得する場合 true,
 *             属性値を取得する場合 false
 * @return キー情報と値がペアになったオブジェクトのリストを返す。
 *         (※)対応するキー情報(タグ構成)が存在しない場合、
 *             要素数が０のリストを返します。
 *             キー情報が存在する場合の要素数は、
 *             テキスト値なら１、属性値なら１以上となります。
 */
private ArrayList<KeyValue> getKeyValues(boolean isText){
    ArrayList<KeyValue> keyValueList = new ArrayList<KeyValue>();
    for(KeyInfo key: keyInfoList){
        if(parser.getDepth() != key.TAG_NAME.length) continue;
        for(int index1 = key.TAG_NAME.length-1; index1 >= 0;
            index1--){
            if(!key.TAG_NAME[index1].equals(tagName[index1])) break;
            if(index1 == 0){
                if(isText && key.ATTRIBUTE_NAME == null){
                    Util.log(INSTANCE, tagName[parser.getDepth()-1]
                        + "="+ parser.getText());
                    keyValueList.add(
                        new KeyValue(key, parser.getText()));
                    return keyValueList;
                }
                for(int index2 = parser.getAttributeCount()-1;
                    index2 >= 0; index2--){
                    if(key.ATTRIBUTE_NAME.equals(
                        parser.getAttributeName(index2))){
                        Util.log(INSTANCE, key.ATTRIBUTE_NAME +"="
                        + parser.getAttributeValue(index2));
                        keyValueList.add(new KeyValue(
                            key, parser.getAttributeValue(index2)));
                    }
                }
            }
        }
    }
    return keyValueList;
}
```

```
    class KeyInfo {
        /** ID番号  */ public final int      ID_NUM;
        /** キー説明 */ public final String   KEY_DESC;
        /** タグ構成 */ public final String[] TAG_NAME;
        /** 属性名  */ public final String   ATTRIBUTE_NAME;
        public KeyInfo(int id, String desc, String[] tag, String attr){
            //属性を取得する場合のコンストラクタ
            ID_NUM         = id;
            KEY_DESC       = desc;
            TAG_NAME       = tag;
            ATTRIBUTE_NAME = attr;
        }
        public KeyInfo(int id, String desc, String[] tag){
            //テキストを取得する場合のコンストラクタ
            ID_NUM         = id;
            KEY_DESC       = desc;
            TAG_NAME       = tag;
            ATTRIBUTE_NAME = null;
        }
    }

    class KeyValue {
        /** キー情報 */ public final KeyInfo key;
        /** 値      */ public final String  value;
        public KeyValue(KeyInfo key, String value){
            this.key   = key;
            this.value = value;
        }
    }
}
```



src/com/cch_lab/android/sample/tinyweatherinfo/ HttpUtil.java (1/1)

HttpClient を利用して、指定された URI からの InputStreamを取得するユーティリティクラス

```
package com.cch_lab.android.sample.tinyweatherinfo;

import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import java.io.InputStream;
import java.io.InputStreamReader;

public class HttpUtil {
    private final Object INSTANCE = this;
    private DefaultHttpClient httpClient         = null;
    private HttpGet           httpGet            = null;
    private HttpResponse      httpResponse       = null;
    private InputStream       httpInStream       = null;
    private InputStreamReader httpInStreamReader = null;

    public HttpUtil(){
        httpClient         = null;
        httpGet            = null;
        httpResponse       = null;
        httpInStream       = null;
        httpInStreamReader = null;
    }

    public void openHttpConnection(String url) {
        try{
            httpClient   = new DefaultHttpClient();
            httpGet      = new HttpGet(url);
            httpResponse = httpClient.execute(httpGet);
        }
        catch(Exception e){
            Util.log(INSTANCE, "InputStream get failed.");
            closeHttpConnection();
        }
    }

    public void closeHttpConnection() {
        try{
            if(httpGet    != null) httpGet.abort();
            if(httpClient != null) httpClient.getConnectionManager()
                                             .shutdown();
        }
        catch(Exception e){
            Util.log(INSTANCE, "HttpConnection close failed.");
        }
        httpResponse = null;
        httpGet      = null;
        httpClient   = null;
    }

    public InputStream openHttpInputStream() {
        try{
            httpInStream = httpResponse.getEntity().getContent();
        }
        catch(Exception e){
            Util.log(INSTANCE, "InputStream open failed.");
            closeHttpInputStream();
        }
        return httpInStream;
    }

    public void closeHttpInputStream() {
        try{
            if(httpInStream != null) httpInStream.close();
        }
        catch(Exception e){
            Util.log(INSTANCE, "InputStream close failed.");
        }
        httpInStream = null;
    }

    public InputStreamReader openHttpInputStreamReader() {
        try{
            openHttpInputStream();
            httpInStreamReader = new InputStreamReader(httpInStream);
        }
        catch(Exception e){
            Util.log(INSTANCE, "InputStreamReader open failed.");
            closeHttpInputStreamReader();
        }
        return httpInStreamReader;
    }

    public void closeHttpInputStreamReader() {
        try{
            if(httpInStreamReader != null) httpInStreamReader.close();
        }
        catch(Exception e){
            Util.log(INSTANCE, "InputStreamReader close failed.");
        }
        closeHttpInputStream();
        httpInStreamReader = null;
    }
}
```

src/com/cch_lab/android/sample/tinyweatherinfo/ Util.java (1/1)

```
package com.cch_lab.android.sample.tinyweatherinfo;

import android.util.Log;

public class Util {
    private static final String MAJOUR_TAG = "TinyWeather";
    public static void log(Object obj, String message){
        Log.d(MAJOUR_TAG+":"+obj.getClass().getSimpleName(), message);
    }
}
```

ログ出力を行うだけの
ユーティリティ・メソッド