始める前に・・・

* http://tnoho.com/and4 へアクセス
* パスワードは shingen
* プロジェクトファイルをダウンロードしてインポートしておいてください

Android開発実践

よく使われるListViewのカスタマイズ

源　拓洋

よく使われるListView

ニュースと天気 (標準)

YouTube (標準)

ヤフオク

ListViewのカスタム

どうやって実装しているのか理解するのが目的です≠実践

サンプルコンテンツは…

リクルートさんのカーセンサーLabから
中古車情報を引っ張ってくる
登録不要でRESTで画像があるWebAPIは、なかなかないw

Content.java

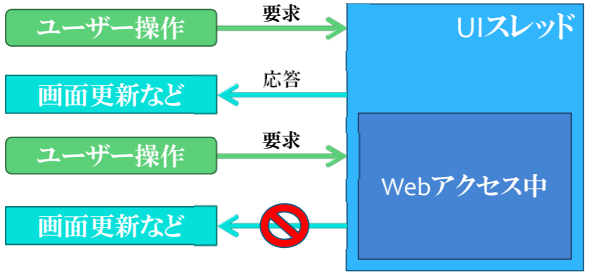
と言うことで、サンプルプログラムでは、
APIドキュメントのUsedCarに応じた
GetterSetterなContentクラスを作ってあります。

カーセンサーラボ.net
http://www.carsensorlab.net/

Price	価格。応談の場合"応談"が入ります
Meter	走行距離(km)。不明の場合"不明"が入ります
Color	色
ImageSUrl	画像リンク（Sサイズ）
ImageLUrl	画像リンク（Lサイズ）
Description	詳細
EntryYear	登録年
Region	広域エリア（北海道、東北～九州、沖縄）
RegionRoman	広域エリア（ローマ字）
Prefecture	都道府県
PrefectureRoman	都道府県（ローマ字）
BodyType	ボディタイプ
DetailURL	詳細URL
DetailMobURL	詳細URL（モバイルサイト用URL）
CatalogURL	カタログURL
CatalogMobURL	カタログURL（モバイルサイト用URL）

Webアクセスする必要がある

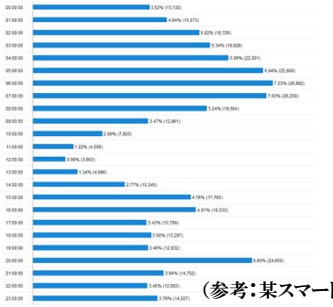
Webアクセスを何も考えずにプログラムすると・・・



Webアクセスにかかりっきりで応答できない

AndroidはUIスレッドから5秒以上応答が無いと ユーザーにダイアログボックスで警告が出てしまう

数Mbpsでる回線に1GHzのCPUなんだから、無視しても…

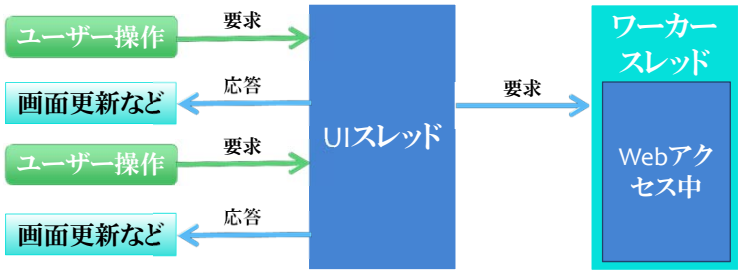


スマートフォンのアクセスは
朝と夕方がスゴイ！

耐えきれず通信速度が
落ちるキャリアが・・・

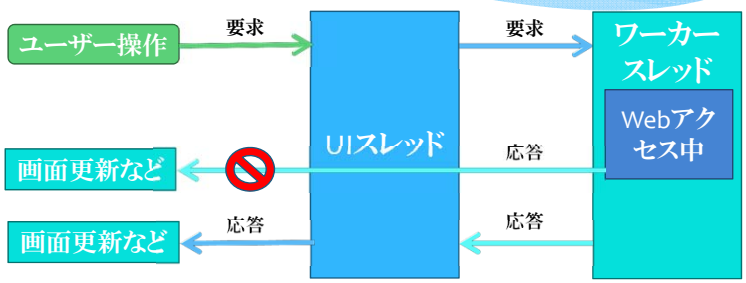
（参考：某スマートフォンコミュニティサイトのデータ）

別スレッド



Webアクセスを他に任せてしまうので、応答が可能になる

別スレッドは画面を変えられない



UIスレッド以外は画面更新などの応答ができないので、
UIスレッドを通す必要がある＝プログラムが複雑に・・・

お手軽別スレッドAsyncTask

threadやhandlerを使わずに（意識せずに）別スレッドで処理を行ってUIスレッドに結果を反映できる便利クラス

AsyncTask内(Protected)のメソッド

onPreExecute()	別スレッド実行前UI処理 （プログレスダイアログの表示など）
doInBackground(Params...)	別スレッド処理 戻り値はonPostExecute(Result)へ
onProgressUpdate(Progress...)	別スレッド進捗状態UI処理 doInBackground(Params...)中に publishProgress(Progress... values)を呼ぶことで 実行される。 （プログレスダイアログ進捗更新など）
onPostExecute(Result)	別スレッド実行後UI処理 （プログレスダイアログの消去など）
publishProgress(Progress...)	doInBackground(Params...)中に使う
onCancelled()	キャンセルされた際のUI処理

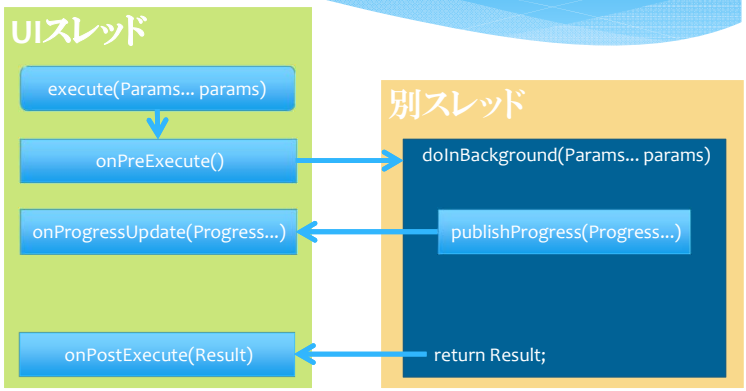
AsyncTask内(Public)のメソッド

execute(Params... params)	実行。引数はdoInBackground(Params...)へ
boolean cancel(boolean mayInterruptIfRunning)	キャンセル。引数trueは中断、falseは実行中の場合完了させる。
getStatus()	ステータス取得、完了。未実行。実行中が得られる
isCancelled()	別スレッド実行後UI処理 （プログレスダイアログの消去など）

でもCancel系は、Android2.3(Gingerbread)以降じゃ無いと動作しないんです。。。

ええ、AndroidOSのバグです。たまにあります。。。

AsyncTaskの処理の流れ



Asynctaskクラスの作成

ListDownloadTask.java

AsyncTask継承クラスを作る

ListDownloadTask **extends AsyncTask<String, Void, String>**

AsyncTask< ①, ②, ③ >の
①はdoInBackground(Params...)のParamの型
②はonProgressUpdate(Progress...)のProgressの型
③はonPostExecute(Result)のResultの型
をそれぞれ記入

doInBackgroundとonPostExecuteをOverride

最低でもこの二つをOverrideしなければならない。
というかこの二つを使わないAsyncTaskって・・・

Asynctaskの実行

MainActivity.java

```
ListDownloadTask listDownloadTask = new ListDownloadTask(this);
listDownloadTask.execute(new  String[]{ "http://tnoho.com/", “123”});
```

AsyncTaskを継承したクラスでexecuteを呼べば実行される。
今回のサンプルでは

ListDownloadTask **extends AsyncTask<String, Void, String>**

としたので、executeの引数はString型

サンプルのAsyncTask継承クラス

ListDownloadTask.java

HttpでXMLをとってきて、XMLを解析し、その結果ArrayList<Content>に詰めて、ListViewに追加しています。

ImageDownloadTask.java

Httpで画像をとってきて、縮小した上で渡されたImageViewに表示したり、Contentに入れたりしています。

画像の縮小にはこんな便利な物を使ってます。

```
public static Bitmap createScaledBitmap
        (Bitmap src, int dstWidth, int dstHeight, boolean filter)
```

処理の内容については私より上手なプロの方が参加者に・・・

ListViewのカスタム

本題・・・のほうが短くなった・・・

UIの準備

main.xml

ListViewだけです。他に何もありません。

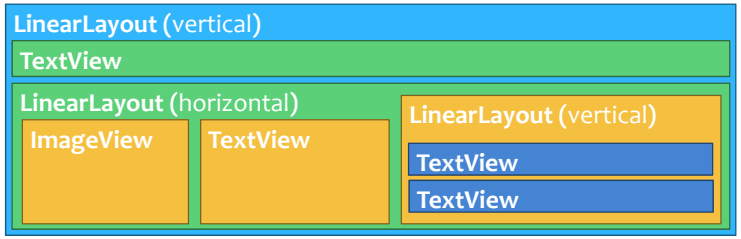
listitem.xml

リスト一行分のレイアウトです。
LinearLayoutを幾重にも重ね合わせ、
右記のような表示を実現しています。

↓常時クルクル回るのとテキストがあるだけ。

footer.xml

Listview.xmlの構成



標準のListViewって？

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1);
String[] list = { "安藤さん", "伊藤さん", "宇藤さん", "江藤さん", "加藤さん"};
for (int i = 0; i < list.length; i++)
{
        adapter.add(list[i]);
}
listView = (ListView) findViewById(R.id.listView1);
listView.setAdapter(adapter);
```

カスタムされたListViewは・・・

MainActivity.java

アイテムの追加を除くと

標準の場合

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1);
ListView  listView = (ListView) findViewById(R.id.listView1);
listView.setAdapter(adapter);
```

カスタムの場合

```
ArrayList<Content>  contents = new ArrayList<Content>();
ContentsAdapter contentsAdapter = new ContentsAdapter(this,
        R.layout.listitem, contents);
ListView  listView = (ListView)this.findViewById(R.id.lv_main);
listView.setAdapter(contentsAdapter);
```

意外にもほとんど同じ

どこが違うん？

MainActivity.java

```
ArrayList<Content>  contents = new ArrayList<Content>();
```

ListDownloadTaskで得たデータを利用するためにContentのArrayListを使ってます。

```
ContentsAdapter contentsAdapter = new ContentsAdapter(this,
        R.layout.listitem, contents);
```

ArrayAdapterが全くの別物に・・・
先ほど出したオリジナルのレイアウトファイルを読み込んでいます。

```
ListView  listView = (ListView)this.findViewById(R.id.lv_main);
listView.setAdapter(contentsAdapter);
```

全く同じ

~~ListView~~ ArrayAdapterのカスタム

ContentsAdapter.java

ArrayAdapter継承クラスを作る

```
ContentsAdapter extends ArrayAdapter<Content>
```

getViewをOverride

getViewは**リストビューに一行を表示しようとする度に呼ばれる**

```
View getView(int position, View convertView, ViewGroup parent)
```

position :表示されようとするAdapterにセットされたデータのposition

convertView :可能であれば古いViewを使い回す。できなければ、null

戻り値にはpositionに対応したViewを渡す。→つまりこいつを変える。

getView!

ContentsAdapter.java

可能であれば古いViewを使い回す。できなければ、null
nullなところには、作ってある一行分のレイアウトを表示したい。というわけで

```
if(v == null){
    v = inflater.inflate(resourceId, null);
}
```

なにこれ。

LayoutInflater

```
this.inflater = LayoutInflater.from(mainActivity);
View v = inflater.inflate(resourceId, null);
```

resourceIdに指定したレイアウトのリソースファイルからViewを一括生成！

getView!!

ContentsAdapter.java

Viewを生成したあとは・・・

```
TextView tvCarDescription = (TextView)v.findViewById(R.id.car_description);
tvCarDescription.setText(content.getStrDescription());
```

よく見覚えのある関数が並びます。でも赤の文字に注目！**V.**がついています。
いままではつけていませんでした。

```
setContentView(R.layout.main);
TextView tvCarDescription = (TextView)this.findViewById(R.id.car_description);
```

実は、**this.**が省略されてついています。今まで呼んでいたのは、
Activityクラスのfindview ById
今回呼んだのは
Viewクラスのfindview Idやっていることは同じでもクラスがちがいます。

`return v;` で、データをsetしたViewを渡します。

getView!!!

ContentsAdapter.java

お、こんなところにImageDownloadTask

```
ImageView ivCarImage = (ImageView)v.findViewById(R.id.car_image);
if(content.getBmpImage() != null){
        ivCarImage.setImageBitmap(content.getBmpImage());
} else {
    ImageDownloadTask task = new ImageDownloadTask(ivCarImage, content);
    task.execute(content.getStrPhotoUrl());
}
```

画像があればImageViewにセット
画像が無いと、ダウンロードしに行きます。

getViewは複数回呼ばれる事もあります。
また、ImageViewを渡してしまっていますが、ImageViewは使い回されるため、
余りよい実装ではありません。今回は簡素化のため、このようにしました。

読み込み中・・・

MainActivity.java

読み込み中・・・の表示もレイアウトファイルを使ったので、LayoutInflaterを使った。

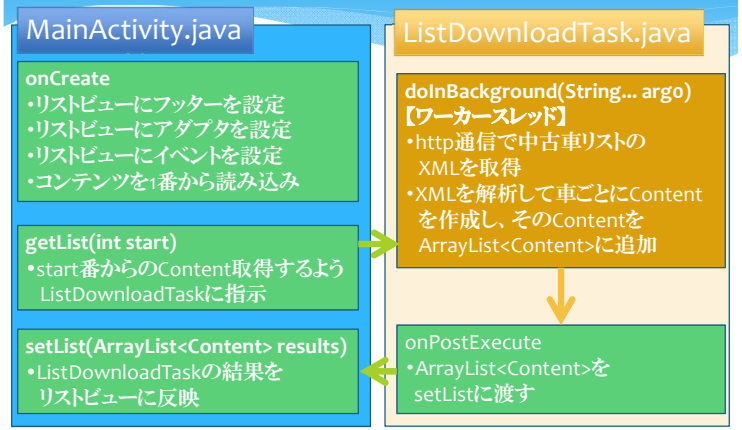
```
viewFooter = LayoutInflater.from(this).inflate(R.layout.footer, null);
listView.addFooterView(viewFooter);

contentsAdapter = new ContentsAdapter(this, R.layout.listitem, contents);
listView.setAdapter(contentsAdapter);
```

ListViewのフッターはListView.addFooterView(View)で表示可能になる。

setAdapterより先に呼んであげないといけないお約束がある。

こういうところで、はまって一日無駄にするのは良くあるので注意。

サンプルアプリ概要その1



リストの取得

MainActivity.java

MainActivityでのリストの読み込みを記述する。

```
public void getList(int start)  {
    if(viewFooter.getVisibility() == View.GONE)
        return;
    ListDownloadTask listDownloadTask = new ListDownloadTask(this);
    listDownloadTask.execute(new String[]{"http://...", Integer.toString(start)});
}
```

ListDownLoadTaskはstartの指定値から20個づつ読み込む。

ところで、View.GONEは一体・・・

```
getList(1);
```

onCreateの一番下には初回読み込みが・・・

リストの追加

MainActivity.java

```
public void setList(ArrayList<Content> results)  {
  if(results.size() < 20)
    viewFooter.setVisibility(View.GONE);
  contents.addAll(results);
  contentsAdapter.notifyDataSetChanged();
}
```

ListDownLoadTaskの別スレッド実行後UI処理onPostExecuteから呼ばれます。
contents.addAll(results);で新規取得20件をAdapterに一括追加。
contentsAdapter.notifyDataSetChanged();で画面表示に反映されます。

ん？新規取得が20件いかで、footerを非表示？
20件以下だとこれ以上コンテンツが無いと見なして非表示にするようです。
と言うことはgetListのは・・・皆さんは、もっとまともなフラグを使いましょう。

コンテンツを利用させて頂いた、カーセンサーのPVに貢献

MainActivity.java

```
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View arg1, int position,long arg3) {
        ListView listView = (ListView) parent;
        Content content = (Content)listView.getItemAtPosition(position);
        Uri uri = Uri.parse(content.getStrUrl());
        Intent intent = new Intent(Intent.ACTION_VIEW,uri);
        startActivity(intent);
    }
});
```

タップしたポジションのデータからURLを取得して、Intentを飛ばせば、
選んだ車の詳細ページに飛ぶことができます。

サンプルアプリ概要その2

MainActivity.java

getView
- 呼ばれた行に応じたContentを取得
- convertViewがnullなら LayoutInflaterでViewを生成
- ViewにContentのデータをセット
- Contentに画像が無ければ、ImageDownloadTaskで画像を取得
- 行にViewをセット

この呼び方は余り良くない。。。

ImageDownloadTask.java

doInBackground(String... argo)
【ワーカースレッド】
- 画像を取得

onPostExecute
- Contentに画像をセット
- ImageViewに画像をセット

無限スクロール

ContentsAdapter.java

表示中リストの下部までスクロールして行ったら、自動的に次を読み込むようにします。
Twitterクライアントなどでよく見かけることがあります。

getViewに

```
if(position == contents.size() - 5)
    mainActivity.getList(contents.size() + 1);
```

を追加。これで残りの未表示が5になった時点で次の読み込みが開始されます

時間は余ってますか？

というわけで、中々に汎用性の高そうなアプリケーションができました。

取得元URLとXMLパーサーを書き換えて別のコンテンツに対応させるもよし。

API仕様とにらめっこして、検索窓を実装するのもよし。

タップしたときブラウザにジャンプでは無く、詳細を表示するActivityを追加するもよし。

私の実装の穴を埋めて完璧を目指すのもよし。

私の説明の穴を埋めるために質問するもよし。