

# 梨好きプログラマの人生ログ

20代後半にプログラマに転身。スマホアプリ開発したり、Webを勉強したり。技術関連メモ、本の紹介、その他趣味など。

2014年7月25日金曜日

## [Android] ViewのonMeasureとかonLayoutについてまとめてみる

## [Android] ViewのonMeasureとかonLayoutについてまとめてみる

Androidアプリを作るとき、よくCustomLayout、CustomViewを作ります。しかしなんとなく作りたいものが作れるけど、onMeasureとかonLayoutとかよく分からないまま使っていました。

折角なので、View周りの仕組みについて細かく調べてみた。

参考にしたのは、Google I/O 2013のWriting Custom Views for Android。それとAndroid DevelopersのReferenceから。

まずは本家Googleさんの説明を学んでみたいと思います。



## Attachment/Detachement

### onAttachedToWindow

YouTubeの4分前後から始まります。Android Developersは[ココ](#)のページ

Referenceだと下記のような説明になっています。

*This is called when the view is attached to a window. At this point it has a Surface and will start drawing. Note that this function is guaranteed to be called before onDraw(android.graphics.Canvas), however it may be called any time before the first onDraw – including before or after onMeasure(int, int).*

訳：ビューがウィンドウにアタッチされた時に呼ばれる。このときビューはSurfaceを持っており、描画が始まる。このメソッドはonDrawの前に呼ばれることを保証しているが、最初のonDrawの前に呼ばれることがある。onMeasureの前後も含まれる。

Google I/Oでは、こんな感じ

- addViewとかした時に親ビューがWindowにアタッチされていれば、まず呼ばれるのがこのメソッド
- View自身がアクティブになるのを知るときがこのタイミング
- いろんなListenerを登録するのはこの時
  - ListViewでは、このタイミングでデータ変更のListenerをRegisterしている

### onDetachedFromWindow

referenceは[ココ](#)。

Google I/Oでは、こちらの方が重要だと言ってますね。

例のごとく、Referenceから引用します。

*This is called when the view is detached from a window. At this point it no longer has a surface for drawing.*

訳：ビューがウィンドウから切り離されたときに呼び出される。この時点では、描画処理はもう走らない。

- ビューがremoveされた時に呼び出される
  - ListViewをスクロールしてセルビューがrecycleされる時、removeViewしたとき、Activityを破棄しているときなど
- このタイミングはいろいろとスケジュールされていたものを終わらせる最適なタイミングである
  - 例えばアニメーション
  - バックグラウンドで走っていたスレッド
  - はたまたdrawing cacheなど、ここで抹消するのが最適であります
  - 特にAndroid 3.0から追加されたHardware Renderer、これはWindowにアタッチされたタイミングで数々のネイティブオブジェクトを生成している。なので、onDetachedFromWindowでクリーンナップする必要がある。なぜかって？コレ以降はViewはそれらを使う必要がいっさいないからさ。的な。
- また、デタッチされた段階でいろいろクリーンされるので、これらのメソッドを呼ぶ場合は、十分に気をつける事。いろんなバグが出るかもしれない。
- なぜ、有名なActivityのライフサイクルのようにViewのライフサイクルというものが存在しないのか
  - Activityはアプリケーションのトップレベルにいるものだが、Viewはそれよりも低い所にいる概念である。そのためViewはActivityのライフサイクルなんか知ったこっちゃない。
  - 自分でCustomViewを作り、listener/callbackを自分で実装しよう。そのcallbackをinvokeするのがActivityの役目である。

## Traversal

YouTubeの9分20秒くらいからです。

Traversal?って何だって感じですが、直訳すると「横断」。よく分かりませんが、この場合、親ビュー-子ビューを横切って処理する事を意味しているのかと思われます。

そんなTraversalのトリガーになるのは3つある。

1. animation
2. requestLayout()
3. invalidate()

気になるならViewRootImpl.javaを見ましょう。GrepCodeとかであるでしょうね。

- traversalのリストが出されたら、measurement、layout、drawingの順番に処理されて行く。
  - この順番は保証されている。
- 複雑なのはmeasurementとlayout。これらは常に一緒についてまわる。君のアプリで変なことをやっていない限りは。

## Measurement And Layout

YouTubeの10分50秒くらいから。

- ViewGroupは子ビューのMeasure、Layoutを必ず行う。
- requestLayout()がすべてのステップの始まりとなる。
  - これによってすべてのビューの再計測が始まる。
- requestLayout()は再帰的な処理を行う。サイズの変更は関連するツリー構造をたどってすべてのビューに関して処理を行う。
  - これが、レイアウトに関してあまり深い階層で組まない方がよい理由でもある。
  - 深い階層の場合、requestLayout()を呼ぶと再帰的に処理が進んでしまうため、場合によってはすごい時間がかかってしまう。この事態を避けたいので、GoogleとしてもUI Toolkitで警告するようにはしているけども。

### onMeasure()

- Viewそれ自身のサイズ及び、すべての子ビューのサイズを決定する。
- Viewのmeasure()メソッドにより呼び出される。
  - 再帰的な処理のため、すべての子ビューにわたってmeasure()が呼び出され、ビューのサイズが決定するのである。
- 計測するためのパラメータはMeasureSpecでまとめられている。
  - makeMeasureSpec()メソッドによって、パラメータがひとまとめにされ、それぞれのgetterメソッドにより取得する事が可能。
- ViewGroupのサイズが計測されると、ViewGroupは子ビューのサイズも計測する責任がある。
- onMeasure()は実際にはreturnがない。なので、setMeasuredDimension()を明示的に呼んで、widthとheightをセットする必要がある。これ忘れたら、frameworkはexceptionを投げますよ。
- ここで設定したwidth、heightはgetMeasuredWidth()、getMeasuredHeight()で取得できる。
  - getWidth()、getHeight()ではないので気をつけましょう。
  - これらのメソッドは、レイアウトが完了するまで値を持たないですよ。

### MeasureSpec

YouTubeの14分後半から。

3つ種類アルヨ。

#### MeasureSpec.EXACTLY

- そのまんま。固定的な値の時に使われる。
  - match\_parent, OODp, layout\_weighty=1など。

#### MeasureSpec.AT\_MOST

- MeasureSpecで設定された値に等しい、もしくはそれ以下のサイズになるよ、ということ。
  - 親Viewのサイズがwrap\_contentの場合、子ビューのサイズを決定するのにAT\_MOSTが使われる。

#### MeasureSpec.UNSPECIFIED

- どんなサイズになるか分かりません。
- ビューは可能な限り無限にサイズを持つ。
- ScrollView、ListViewなんか典型的な例ですね。

#### getChildMeasureSpec()

適切なMeasureを得るためのフレームワークの一つがこれ。

- 例
  - parentのレイアウトのサイズが300pxの場合、で子ビューがwrap\_contentのとき。
    - parentビューが小ビューに送る情報は、AT\_MOSTモードで最大300px

### onLayout()

YouTubeの17分くらいから。

- Measurementが完了したら、次は子ビューたちのposition(位置)を決定させる。
- そう、この時によくあるoffset、margin、padding等が適用されるのである。
- traversalの途中で一回呼ばれるのがこのメソッド。
  - measureの段階だと、何回も呼ばれてアレなので、重い処理などはここで行うのが適切である。
- サイズに関しては、onMeasureで既に計算がされている。なので、サイズを取得したい時はここでgetMeasuredWidth()、getMeasuredHeight()を呼ぶといいよ。

### LayoutParams

YouTubeの20分くらいから。

XMLの属性でよく定義されるLayoutParamsですが、layoutに関連しているstaticなクラスです。

- 一番基本的なクラスはViewGroup.LayoutParams。
- その他にもLinearLayout.LayoutParamsやFrameLayout.LayoutParamsなどもある
  - これらを使わなくても基本的なLayoutParamsを使っていればいいよ。
- その他に便利なものがMarginLayoutParams。
- LayoutParamsはXMLでもインポートする事ができる。
- レイアウトに設定できるもので、widthやheightはおなじみですが、LinearLayoutの場合はweightというものもある。
- withやPheightの他に必要があってCustom LayoutParamsを作ったなら、4つのメソッドを使う事ができるよ。
  - checkLayoutParams:
    - ViewにAddした時に呼び出される
    - そのLayoutParamsがViewGroupのものと合致しているかをチェックしている。合っていないとExceptionを投げちゃうよ。
    - LayoutをAddしたとき、前のViewがFrameLayoutだったときはFrameLayout.LayoutParamsが正解
  - generateLayoutParams(LayoutParams)
  - generateLayoutParams(AttributSet)
  - generateDefaultLayoutParams
    - これらはReference参照
- onMeasureとかonLayoutで重い処理をやっているなら、LayoutParams使ってみるといいですよ。RelativeLayout.javaなんかすごい参考になる。

### Draw

YouTubeの22分40秒くらいから。

- invalidate()によって呼び出される。
  - invalidateChild()が再帰的に親ビューに向かって呼び出される。
- これは非常に重い処理である。requestLayout()は単にフラグを処理しているだけだが、invalidate()はアニメーションや様々な描画処理が走る。
- onDraw()はOverrideすることができ。
- setWillNotDraw()というメソッドがあり、引数にtrueを指定すると描画処理をスキップする事ができる。子ビューは描画されるが、これを指定するとonDraw()が呼ばれずに、代わりにdispatchDraw()というものが呼ばれるよ。親ビューの描画処理はされずに、子ビューたちの描画を行うようになる。
- drawChild()メソッドによって、トランスフォーム、アニメーション等の描画処理が子ビューに対して処理される。これもOverrideできるので、すべての子ビューに対して呼んであげると良い

なんか非常に長くなりました。最後なんかもう疲れて飛べなくなりましたが、とりあえず、Androidのソースコード読むといういろいろわりそうですね。またそのうち、レポートします。

投稿者 dog man 時刻: 22:12



ラベル: android, attach, detach, onDraw, onLayout, onMeasure, view, ビュー, レイアウト, 描画

0件のコメント:

コメントを投稿

コメントを入力...

コメントの記入者: **okuda0715Tech**

公開

プレビュー

☐ お知らせを受け取る

このブログを検索

検索

amazon



自己紹介

**dog man**  
詳細プロフィールを表示

ブログアーカイブ

- 2015 (2)
- 2014 (18)
  - 9月 (3)
  - ▼ 7月 (11)
    - [Android] ViewのonMeasureとかonLayoutについてまとめてみる
    - [Android] Android StudioにSupport Library v4を設定する
    - [趣味] BOSE 201V& DENON PHE 390SE使ってみた
    - [Android] Fragment(簡単なサンプル
    - [読書] 社長失格読んでみた
    - Bloggerでテンプレートを変更する
    - と、Synthavhighlighterが動かなくなる
    - [趣味] おすすめイヤホン
    - Happy Hacking Keyboard
    - [Android] Fragmentを使ってみる
    - [読書] 孫子の兵法
    - Android開発で永遠にミュレータを使う(iOSのミュレータに匹敵)
- 4月 (2)
- 3月 (2)
- 2013 (13)

人気の投稿

[Android] ViewのonMeasureとかonLayoutについてまとめてみる  
[Android] ViewのonMeasureとかonLayoutについてまとめてみる  
Androidアプリを作るとき、よくCustomLayout、CustomViewを作ります。しかしなんとなく作りたいものが作れるけど、onMeasureとかonLayoutとかよく...

[Android] SQLiteのinsert時を比較してみた(高速化)

Android開発でよくSQLiteを使うけど、使いようによって速度が劇的に早くなる。ってのは知ってたけど実際に試してみたら、①普通にSQLiteDatabase #insertを使う ②上のやつにトランザクションを使う ③PrecompileS...

[Android] Android StudioにSupport Library v4を設定する

Android Studioを使って間もないのですが、Eclipseとはどうも使い方が違っていたので、Eclipseだと、Add Support Libraryとか、他のライブラリだとビルドパスとかでいじっていたけど、Android Studioだと、build.g...

Android StudioでPlease Specify Android SDKと出てSDK Managerが開かない

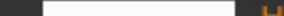
さっきAndroid Studioを0.2.4にアップデートしたときに、SDK Managerを開いたらPlease specify android sdkというエラーが出て前になくなった。そんなときはFile -> Project S...

[Android] SQLite transaction中にselectする(同時に)

ハマりました。やりたいことは、単純にbeginTransaction -> setTransactionSuccessful()の間に、queryでデータを読み出すだけ。IMMEDIATEモードを使えば出来るはずですが、beginTransaction NonExclusive(...

[Android] Fragment(簡単なサンプル

試しにフラグメントをつかって画面遷移ためしてみた。メインのActivity 基本的にActivityはこれ一つでOK。replace()でスタックとみなすフラグメントを表示し、addToBackStack()で現在のフラグメントをバックスタックにまわす。これを行...



Hacking Keyboard

キーボードなんて何でも同じだよ、なんて思っているアナタ。違います！プログラマはキーボードとお友達！なんです。使ってみて初めて分かるこの感覚。いろんな人に見られてほしくない。おなじに筆者が使ってる来たキーボードは...付属のキーボードはMajestou...

[読書] ハッカーのたのしみ

「ハッカーのたのしみ」本物のプログラマはいかにして問題を解くか」ほんとの書名は「Hacker's Delight」なんだけど、読んでみた本だけど、まだ全然読めてない。Amazon.comとこのレビューでも、★★★★★ なんてすごい本。ただ、ワタ...

androidでMVC

アプリちょっと複雑になってくるともうぐちゃぐちゃ。なんてことあるせいで自分だけではないかも。xml情報をモデルに変換し、画面は画面でいろいろのパーツがあったり、DB連携があったり。そうです、いわゆるMVCアーキテクチャというものの大切さを最近思い知ったのです...

[読書] 孫子の兵法

読んでいます。と、ル・ケイツや孫さんなど、世界のトップ経営者のバイブルと呼ばれる孫子の兵法。中国春秋時代に、孫武が作ったとされる、戦争に勝つための兵法書。今の時代、実際に戦争で戦うという事はないけど、ビジネスの世界は戦争と同じで強い者が生き残り、弱い者は自然と社会...