شرح الـ LayoutInflater وفهم الـ attachToRoot parameter

haythem.kmal **في ٣٠ مارس**

السلام عليكم مبرمجي المستقبل
كنت أظن أن عهدي بالشروحات انتهى لكن ماذا أفعل وقد دفعني أخي رضى **@tabsit1999** بسؤاله عن ماهية هذا السطر البرمجي

```
listitemview = LayoutInflater.from(getContext()).inflate(R.layout.item_inside_list
```

ولأن أخي رضى عزيز على قلبي فقد أبيت الا أن أطرح اجابة هذا التساؤل كموضوع خاص 😉 وأيضا ليستفيد أصدقائي المبرمجين.

حسنا لنبدأ .
أولا كما تعلمون جميعا أن هذا الأمر يقوم بتحويل ملف الـ xml layout الى View Object بمايحتويه من Viewsو Widgets ايا كانت نوعها. لنشبه هذا الأمر بنفخ البالون (هذا التشبيه مأخوذ من أحد المواقع ولكني لا أعلم تحديدا عنوان الموقع لأن هذا حدث منذ فترة ليست بالقليلة) فلنفترض ان ملف الـ xml layout هو عبارة عن بالون غير منتفخ مرسوم عليه بعض الرسومات وكلنا نعلم أن هذا النوع من البالونات عندما يتم نفخه تظهر روعة وجمال الصور المرسومة عليه . هذا هو الحال مع الـ xml layout وهذا المنفاخ الذي يسمى LayoutInflater . لاحظ انه وكما أنوه دائما أن مطوري الاندرويد دائما مايستخدمون كلمات تصف الى حد كبير الفعل المنوط بهذه الكلمة . كلمة `inflate` تعني نفخ 😉 وكلمة LayoutInflater تعني منفاخ الـ xml layout .
حسنا لنعد الى مايحدث بصفة عامة اذن نحن لدينا ملف xml layout ربما يحتوي على LinearLayout مثلا وبعض الـ TextViews مثلا و و الخ . هذا البالون (xml layout) طالما لم يتم نفخه فهو غير صالح للعرض .أليس كذلك ؟ من الذي يقوم بنفخ هذا البالون أي من الذي يقوم بتحويل الأكواد المكتوبة في صيغة xml layout الى الـ Views objects التي يحتويها؟! أحسنت انه الـ LayoutInflator

حسنا دعنا نفصص هذا السطر البرمجي ولاحظ معي أني سأقوم بالتفصيص الفعلي لهذا السطر
حسنا في البداية كما شرحت نحتاج الى المنفاخ 😂 المسؤول عن عملية النفخ (تحويل الـ xml layout الى Views and widgets)
هناك object يسمى LayoutInflator وانت قمت بانشاءه من خلال هذا السطر البرمجي ولكنك لم تلاحظه بسبب الـ dot notation المتتابع ... في كل الاحوال لنقم بانشاء الـ LayoutInflator

```
LayoutInflater myInflater = LayoutInflater.from(context);
```

أرجوك لاحظ أني فقط أقوم بتفصيص السطر الاتي

```
listitemview = LayoutInflater.from(Context).inflate(R.layout.item_inside_listview
```

فهذا الجزء `LayoutInflater.from(Context)` ماهو الا عبارة عن `LayoutInflater object`
وهو الذي يحتوي على الأمر inflate . وهذا شيء منطقي فطبيعي أن المنفاخ (LayoutInflater) هوا الذي يقوم بالنفخ (inflate) 😂 أي تحويل الـ xml layout file الى Views و widgets مجموعة في View object.

حسنا الان فهمنا ماطبيعة الـ inflate method ومن أين جاءت وماذا تفعل . اذن دعنا نرى ماهي الـ parameters التي تستقبلها هذه الـ method

```
inflate (int resource,
                ViewGroup root,
                boolean attachToRoot)
```

(بداية اذا لم تفهم هذه الـ Parameters فما عليك الا المتابعة مع المثال الذي سأضربه في النهاية وباذن الله سيتضح لديك المفهوم بشكل كامل)
** اولparameter وكشيء منطقي ستحتاج الى الـ xml layout فكما قلنا في النهاية مانريد أن نقوم به هو تحويل xml layout file الى View object أليس هذا هو المقصود ؟!
~~{param}~~

** ثاني parameter هو الـ root وهو عبارة عن ViewGroup وهذا أيضا شيء منطقي ~~{فلنفترض أن ملف الـ xml به عدة TextViews و بعض الـ buttons وياحبذا لو قليل من الـ ImageViews 😄 السؤال المنطقي بالله عليك كيف يمكن ترتيب كل هذه الـ Views ؟؟؟؟؟ }~~
الميثود التي تسمى inflate تحتاج الى نوع من التوجيه وكلنا نعلم أن الـ ViewGroups (مثلا الـ LinearLayout أو الـ RelativeLayout) هي المسؤولة عن ترتيب وضع واماكن الـ views ولذلك نحتاج في ثاني parameter الى أي ViewGroup يقوم بتوجيه الأمر inflate الى كيفية ترتيب أماكن الـ Views و الـ widgets في النهاية داخل الـ View object الذي سيقوم بارجاعه لنا.
أحب أن أرفق هنا تعليق خاص بالمهندس عمر **@OmarAlbelbaisy** يوضح الفقرة السابقة بشكل أعمق ويبين أوجه قصورها(كان لابد من الاضافة هنا لأنها توضح قصور في الشرح والمفهوم أما بالنسبة لباقي التعليق فهو بالأسفل لكني أفضل لك قراءة التعليق بعد قراءة الموضوع حيث يقوم المهندس عمر بزيادة ايضاح بعض الأمور بشكل أكثر عمقا 🙂 . جزاه الله عنا كل خير 😍🌹

> OmarAlbelbaisy:
>
> ثانياً: بخصوص التالي
>
> haythem.kmal:
>
> فلنفترض أن ملف الـ xml به عدة TextViews و بعض الـ buttons وياحبذا لو قليل من الـImageViews 😄 السؤال المنطقي بالله عليك كيف يمكن ترتيب كل هذه الـ Views
>
> فعلياً لا يمكن إضافة أكثر من View في ملف XML دون وجود root layout مباشر يحتويهم في نفس ملف الXML لذلك فائدة الparent ستكون هنا تحديد الlayout params الخاصة بالroot layout الذي يحتوي هذه الviews ولن يكون مسؤولا عن تحديد أماكن وترتيب الviews بشكل مباشر.
>
> الحالات التي سيكون فيها الparent مسؤولاً عن ترتيب مكان الviews هي أن يكون الlayout يحتوي على view واحد كما ذكرت في المثال أعلاه أو أن يتم استخدام وسم merge بدلا من الroot layout وذلك ليتم إضافة الviews إلى الparent layout الذي تم تمريره إلى الinflate method مباشرة ولكن في هذه الحالة الأخيرة سيكون استخدام parent layout وتمرير true كقيمة للparameter المسمى attachToRoot إجباريا وإلا سيحدث خطأ أثناء عملية الinflation.

لعلك الان الى حد ما تكونت لديك فكرة عن فعل هذا الأمر عندما استخدمناه داخل الـ getView method الخاصة بالـ ArrayAdapter .
** نأتي لثالث Parameter وهو مايسبب مشكلة لدى الكثيرين ألا وهو الـ attachToRoot وهو من نوع boolean . نحن نعلم كما ذكرت سابقا أن هذه الـ method ستقوم في النهاية بارجاع View و قلنا أيضا أنه يتم استخدام ViewGroup كنوع من التوجيه . حسنا بالنسبة لهذا الـ parameter اذا قمنا بتعيين قيمته true فان الـ View الذي سيقوم هذا الأمر بارجاعه لك سيكون عبارة عن ViewGroup + Views of the xml layout أي عبارة عن الـ ViewGroup الذي قمنا باستخدامه وتم اضافة له الـ Views الخاصة بال xml layout بما تحتويها من TextViews مثلا أو غيرها .اما اذا تم تعيين القيمة false فانه سيقوم بارجاع الـ View object (المحول من الـ xml layo)بدون اضافته للـ ViewGroup وفي هذه الحالة يتم استخدام الـ ViewGroup فقط كنوع من التوجيه كما شرحت سابقا.

لنعطي مثالا عمليا يشرح هذه القصة
في البداية سنقوم بفتح مشروعا جديدا
طبعا سيتكون لديك MainActivity.java وأيضا الـ activity_main.xml
سنقوم باضافة ملف xml layout (هذا الملف هو من سنقوم بتحويله باستخدام الأمر inflate عن طريق LayoutInflater) سنقوم بتسمية هذا الملف layout_inflate.xml
سيكون بالشكل الاتي

```
<?xml version="1.0" encoding="utf-8"?>
<Button
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="my Inflated Button">

</Button>
```

اذن هو ملف xml بسيط يحتوي على view واحد عبارة عن Button . سنرى بعد قليل كيفية نفخ هذا الـ xml layout file لنحصل في النهاية على View يحتوي على Button أو ربما على View يحتوي على الـViewGroup مضافا اليه هذا الـ Button 😉

ملف الـ activity_main.xml سيكون بالشكل الاتي

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/ParentView"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.android.chat.MainActivity"
    android:orientation="vertical">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Parent Button !"

        />

</LinearLayout>
```

ملف الـ MainActivity.java سيكون بالشكل الاتي

```
package com.example.android.chat;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

// هنا سأقوم بوضع حالات مختلفة لتوضيح الـــ parameters

    }
}
```

حسنا قم بتشغيل المشروع ستلاحظ كما توقعت بالضبط ظهور زرار خاص بالـ main_activity.xml مكتوب عليه Parent Button .

سأقوم الان بتحويل الـ layout_inflate.xml الى View بعدة حالات ولاحظ معي كل حالة

**** أولا حالة**

قم بلصق الكود الاتي أسفل setContentView(R.layout.activity_main);

```
// Cooment:   هذا ما أحتاجه كـــــparameterثاني حيث سأستخدمه كمرشد

LinearLayout myParentView = findViewById(R.id.ParentView);
```

```
// Comment:inflate method لـــ استخدام أستطيع لكي LayoutInflater object ال هوا ا
LayoutInflater myInflater = LayoutInflater.from(this);

View myInflatedView = myInflater.inflate(R.layout.layout_inflate , myParentView, fa
```

حسنا بالنسبة لهذا السطر

```
View myInflatedView = myInflater.inflate(R.layout.layout_inflate , myParentView, fa
```

نقوم هنا بعمل متغير من نوع View اسمه myInflatedView فالأمر inflate سيقوم في النهاية بارجاع View Object ونقوم باستخدام الـ myInflater لاستدعاء الدالة inflate ونقوم بتمرير الـ layout_inflate.xml الذي يحتوي على Button وحيد ونقوم باستخدام الـ myParentView والذي هو عبارة عن الـ LinearLayout في الـ main_activity.xml وفي نفس الوقت نقوم بتمرير القيمة false اذن نحن نقول للأمر inflate من فضلك قم بتحويل الـ layout_inflate.xml الى View واستخدم الـ myParentView (linearlayout) كمرشد ومن فضلك لا تقم باضافة هذا الـ View الى هذا المرشد فقط قم بارجاع الـ View الموجود كما هو في الـ layout_inflate.xml بعد تحويله.
حسنا لنقم باضافة الأمر التالي

```
myParentView.addView(myInflatedView);
```

ثم قم بتشغيل البرنامج ستلاحظ اضافة الـ Button كـ child لل myParentView (linearlayout) أي سترى اثنين Buttons احدهما وهو الأول خاص بالـ main_activity.xml مكتوب عليه Parent Button والاخر يقع أسفل منه خاص بالـ layout_infalte.xml مكتوب عليه my Inflated Button وهو عبارة عن الـ myInflatedView الذي أرجعه لنا الأمر inflate

حسنا قم بمسح الأمر السابق وضع بدلا منه هذا الأمر فقط لكي ترى بعينك ماهو هذا الـ View الذي قام inflate بارجاعه لك

```
setContentView(myInflatedView);
```

ستلاحظ الان ان ماسيظهر لك هو الزر الخاص بالـ layout_inflate.xml مكتوب فيه my Inflated Button . هل اتضحت الرؤية الان .

**** ثاني حالة**
حسنا لنقم بتغيير false الى true كالتالي

```
View myInflatedView = myInflater.inflate(R.layout.layout_inflate , myParentView, tr
```

;كما شرحنا وفي الاعادة افادة نقوم هنا بعمل متغير من نوع View اسمه myInflatedView فالأمر inflate سيقوم في النهاية بارجاع View Object ونقوم باستخدام الـ myInflater لاستدعاء الدالة inflate ونقوم بتمرير الـ layout_inflate.xml الذي يحتوي على Button وحيد ونقوم باستخدام الـ myParentView والذي هو عبارة عن الـ LinearLayout في الـ main_activity.xml وفي نفس الوقت نقوم بتمرير القيمة true اذن نحن نقول للأمر inflate من فضلك قم بتحويل الـ layout_inflate.xml الى View وأيضا من فضلك قم باضافة هذا الـ View الى هذا myParentView (linearlayout) ثم قم بارجاع الـ View بحيث يكون الـ myParentView (linearlayout) مضافا اليه الـ Button View الخاص بالـ layout_inflate.

ثم قم بتشغيل البرنامج ستلاحظ ظهور الاثنين Buttons احدهما وهو الأول خاص بالـ main_activity.xml مكتوب عليه Parent Button والاخر يقع أسفل منه خاص بالـ layout_infalte.xml مكتوب عليه my Inflated Button

بعد ذلك عزيزي المبرمج قم بوضع السطر الاتي بعد ذلك لكي ترى بعينك ماهو هذا الـ View الذي قام inflate بارجاعه لك

```
setContentView(myInflatedView);
```

ثم قم بتشغيل البرنامج ستلاحظ أنه نفس المظهر السابق لأن الـ View الذي قام الـ inflate method بارجاعه لك عبارة عن LinearLayout(with all its child) + Button View attached to this linearLayout. وهذا ماتفعله القيمة true.

حسنا قم بمسح الامر السابق و لنقم باضافة هذا الـ View كما فعلنا في أول حالة عن طريق الأمر

```
myParentView.addView(myInflatedView);
```

هل لاحظت توقف البرنامج ؟! لأن ماتقوم بفعله هو اضافةViews موجودة أصلا داخل ال myParentView.

ملحوظة جانبية
بالنسبة لاخر حالة لعل هذا مايحدث داخل الـ `getView method` اذا قمت بتمرير القيمة true بدلا من false>
فالـ ListView يقوم أصلا باضافة الـ Views الموجودة داخل الـ convertView بعد ذلك.فاذا قمت انت بتعيين القيمة true فكما سبق وأن شرحنا كأننا نقول أننا نريد اضافة هذه الـ Views كـ child داخل الـ ListView (في هذه الحالة الـ ListView هو الـ ViewGroup فهو في حالتنا هذه يقوم مقام الـ LinearLayout في الشرح) ولكن مهلا في الاندرويد لايمكن اضافة أي View مرتين ولأن الـ ListView أصلا يقوم بهذه الاضافة نيابة عنا فلا نستطيع ان نعين الparameter الاخير بـ true داخل الأمر getView والا سيحدث خطأ وسيتوقف البرنامج.

- بما ان ده اخر شرح لي غالبا وغالبا برضه اخر عهدي بيكم كمنتدى فأحب أعبر عن امتناني لوجودي وسطكم ورغم اني كنت ضد فكرة انهم يخلوا المشاركة عليها نوع من التقييم الا ان فعلا تقريبا دي احسن حاجة في المبادرة دي .عرفت ناس على الاقل ظاهريا بتحب الخير لغيرها بجد وفعلا حسيستوني اننا كبلاد عربية مفيش حاجة بتفرق بينا. الأمة دي عظيمة بشعوبها والله بس مش عارفين يستغلونا .ماتعيطش استنى 😂
المهم من الاخر كده اللي حابب يكون على تواصل معايا ياريت يبعتلي على الخاص:) بس خلاص 😁