Android各种Layout布局

Last edited by **caoquanli** 1 month ago

Android各种Layout布局

Table of Contents

布局说明

LinearLayout（线性布局）

布局特点：

- 在水平或者垂直方向上依次按照顺序来排列子元素，控件的排列顺序遵循其在布局文件中被写出的先后顺序。LinearLayout中存在weight这一属性，在水平方向上代表列宽，在垂直方向上代表行距。一般这个值越大，则所占用的控件比例越大。

适用场景：

- 适用于横向或纵向顺序排列的页面，如水平标题栏、纵向列表等。

缺点：

- 对于复杂、内部关联较多的页面，可能会产生嵌套层级过多的问题；
- 对于复杂、内部关联较多的页面，可能会由于大量频繁使用weight导致性能降低。

RelativeLayout（相对布局）

布局特点：

- 以某一个元素为参照物，其余元素均按照其与参照物间相对位置来完成布局。

适用场景：

- 适用于较为复杂，内部关联较多的页面；
- 其属性很好地弥补了LinearLayout的缺点，避免由于页面的复杂导致的嵌套层级过多和大量频繁使用weight导致的性能降低。

缺点：

- 在修改了某一个控件的属性时，其他依赖于该控件的其他控件可能都需要修改其属性；
- 有嵌套层数的限制。

FrameLayout（帧布局）

布局特点：

- 所有的View都会放在左上角，并且后添加进去的View会覆盖之前放进去的View。

适用场景：

- 适用于有全布局切换或多层View重叠的页面，如浏览单张图片、多插页多Tab切换等。

缺点：

- 对于复杂、内部关联较多的页面，实现其布局会比较困难麻烦。

AbsoluteLayout（绝对布局）

布局特点：

- 绝对布局也叫坐标布局，通过指定元素在屏幕上的绝对位置来确定布局，同时需要精确指定控件尺寸。由于页面的适应性和屏幕的兼容性很差，一般很少用到，目前在API中已是弃用状态。

适用场景：

- 适用于屏幕尺寸单一不变，控件按坐标位置精确定位，控件尺寸可精确指定的页面。

缺点：

- 页面的适应性和屏幕的兼容性差；
- 控件尺寸位置需要精确指定；
- 目前在API中已是弃用状态。

TableLayout（表格布局）

布局特点：

- TableLayout是LinearLayout的子类。TableLayout都是由一个或多个TableRow组成的，一个TableRow就代表TableLayout的一行。每个TableRow容器中可以有多个控件，控件个数决定这一行的列数。

适用场景：

- 适用于类似表格形式、从上到下按行排列，每行多列的页面。

缺点：

- 适用场景限制较大，对于适用场景以外的页面较难实现。

GridLayout（网格布局）

布局特点：

- GridLayout布局是Android4.0（API Level 14）新引入的网格矩阵形式的布局控件。容器中的各组件呈M行×N列的网格状分布。每列宽度相同，每行高度相同，各组件的排列方式为：从上到下，从左到右。

适用场景：

- 适用于网格形式的页面。

缺点：

- 适用场景限制较大，对于适用场景以外的页面较难实现。

ConstraintLayout（约束布局）

布局特点：

- ConstraintLayout属于Android Studio 2.2的新特性，是Google在2016年的I/O大会上重点宣传的一个功能。ConstraintLayout可以有效地解决布局嵌套过多的问题。ConstraintLayout是使用约束的方式来指定各个控件的位置和关系的，它有点类似于RelativeLayout，但远比RelativeLayout要更强大。

适用场景：

- 适用于较为复杂，内部关联较多、嵌套较多的页面，可以有效地解决布局嵌套过多的问题。

缺点：

- 由于ConstraintLayout是使用约束的方式来指定各个控件的位置和关系的，所以处理较为简单的页面时，可能比FrameLayout、LinearLayout之类的基础布局类型耗时长。

布局效率

布局效率，主要从measure、layout、draw三部分过程的耗时来进行对比。 比较了 “10层Layout嵌套，加载刷新1000次” 和 “15层Layout嵌套，加载刷新1次” 两种情况。

各Layout耗时比较

- 在measure部分耗时 FrameLayout ≈ LinearLayout ≈ AbsoluteLayout ≈ TableLayout > GridLayout > ConstraintLayout > RelativeLayout
- 在layout部分耗时 所有Layout基本都差不多
- 在draw部分耗时 FrameLayout ≈ LinearLayout ≈ AbsoluteLayout ≈ TableLayout ≈ GridLayout ≈ RelativeLayout > ConstraintLayout
- 总耗时 FrameLayout ≈ LinearLayout ≈ AbsoluteLayout ≈ TableLayout > GridLayout > ConstraintLayout > RelativeLayout

布局效率比较

根据各Layout的耗时比较， * FrameLayout、LinearLayout、AbsoluteLayout、TableLayout属于基础布局类型，布局效率较高； * GridLayout属于特殊布局类型，布局效率中等； * ConstraintLayout、RelativeLayout属于复杂布局类型，布局效率稍低； * ConstraintLayout的布局效率高于RelativeLayout。

综述

各Layout情况

- FrameLayout、LinearLayout、AbsoluteLayout、TableLayout属于基础布局类型，布局效率较高；
- GridLayout使用场景有较大限制，布局效率中等；
- ConstraintLayout、RelativeLayout布局效率稍低，其中ConstraintLayout的布局效率高于RelativeLayout；
- FrameLayout、LinearLayout实现复杂页面比较困难麻烦；
- TableLayout使用场景有较大限制；
- ConstraintLayout、RelativeLayout适用于关联较多、较为复杂的页面；
- RelativeLayout有嵌套层数的限制；
- AbsoluteLayout的页面适应性和屏幕兼容性较差且Android API已弃用；

总结

- 对于较为简单，内部关联较少，嵌套层级较少的页面，推荐使用FrameLayout和LinearLayout；
- 对于较为复杂，内部关联较多，嵌套层级较多的页面，推荐使用ConstraintLayout；
- 使用场景特定的表格、网格型页面，可考虑使用TableLayout、GridLayout；
- 由于页面适应性和屏幕兼容性较差且Android API已弃用等原因，尽量不要使用AbsoluteLayout。