Android. Различия в разметках FrameLayout, RelativeLayout, LinearLayout и TableLayout

Рассмотрим принципиальные отличительные черты четырех типов разметки в Android: FrameLayout, RelativeLayout, LinearLayout и TableLayout.

**FrameLayout**

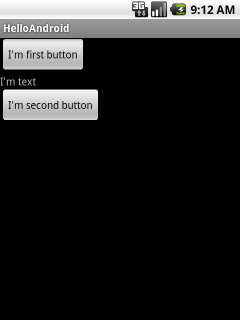
**FrameLayout** - самый простой тип разметки. Может содержать только один дочерний элемент типа View или ViewGroup, который будет выравниваться по верхней левой границе. При добавлении последующих элементов они будут перекрывать друг друга. Пример:

FrameLayout

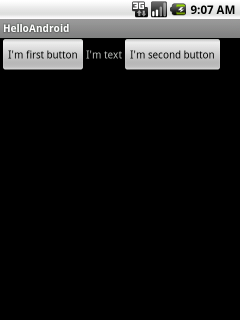
**LinearLayout**

**LinearLayout** в отличие от FrameLayout может содержать больше одного элемента. Как следует из названия, LinearLayout является линейной разметкой, дочерние элементы которой размещаются друг за другом. Может размещать элементы как горизонтально, так и вертикально. При вертикальном размещении каждый элемент занимает отдельную строку, а при горизонтальном - элементы размещаются в одной строке, причем высота строки является высотой самого высокого дочернего элемента. Пример LinearLayout разметки с вертикальной ориентацией:

LinearLayout (horizontal)

  
Рис 1. LinearLayout. Вертикальная ориентация

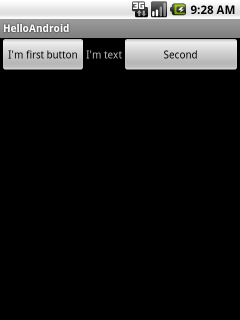
Чтобы получить горизонтальное размещение, достаточно изменить атрибут android:orientation, присвоив ему значение horizontal:

  
Рис 2. LinearLayout. Горизонтальная ориентация

Если Вы хотите, чтобы последний элемент "заливал" оставшееся на строке пространство, необходимо изменить его атрибут android:layout\_width, присвоив значение: fill\_parent:

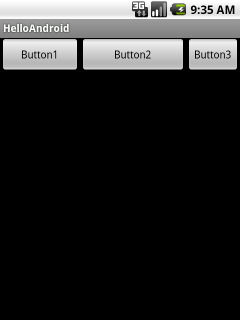
Horizontal LinearLayout с использованием fill\_parent

Результат выполнения этого примера показан на рис. 3:

  
Рис 3. LinearLayout. Горизонтальная ориентация с использованием fill\_parent

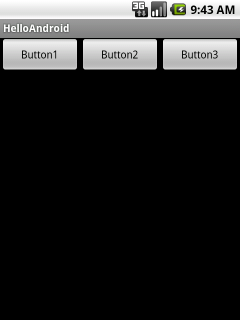
Рассмотрим еще одно широко используемое свойство - layout\_weight ("вес размещения") на следующем примере:

Horizontal LinearLayout с использованием layout\_weight

  
Рис 4. LinearLayout. Горизонтальная ориентация с использованием layout\_weight

Как видно с Рис. 4, наибольшее пространство занимает Button2 из-за того, что имеет самый высокий layout\_weight="2" среди остальных элементов. Button1 имеет вес 1 и поэтому занимает немного меньше пространства, чем Button2. Кнопка с самым низким весом, Button3, занимает меньше всего пространства.

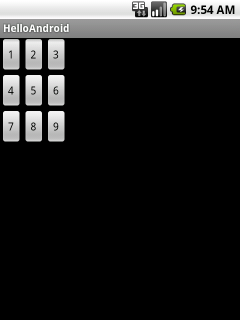
Если бы кнопки Button1 и Button2 имели одинаковый вес, то занимали бы одинаковое пространство. А вот так располагаются кнопки в случае одинакового значения layout\_weight:

  
Рис 5. LinearLayout. Горизонтальная ориентация с одинаковым layout\_weight

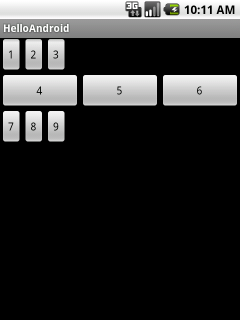
**TableLayout**

**TableLayout** позволяет создавать разметку по подобию таблицы в html. Строка таблицы, TableRow, может не иметь ячеек или иметь несколько ячеек. Допускается также ячейки типа ViewGroup. То есть, ячейка может быть другим типом разметки, в том числе и TableLayout. Пример TableLayout:

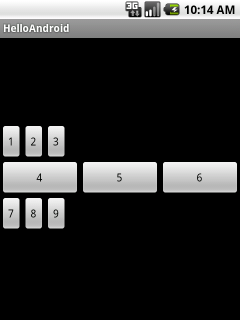
TableLayout

  
Рис 6. TableLayout

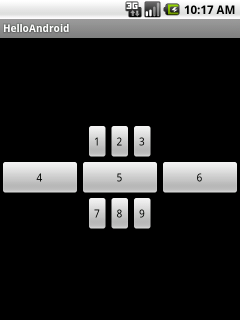
Для дочерних элементов так же как и в LinearLayout может использоваться свойство layout\_weight. Изменим следующим образом пример: установим атрибут android:layout\_weight="1" для элементов btn4, btn5 и btn6:

  
Рис 7. TableLayout с использованием layout\_weight

Чтобы выровнять таблицу вертикально по центру, необходимо добавить атрибут android:gravity="center" в TableLayout:

  
Рис 8. TableLayout с gravity=center

А чтобы выровнять по центру каждую строчку достаточно добавить атрибут android:gravity="center" во все три TableRow:

  
Рис 9. TableLayout с атрибутом gravity в TableRow

Исходный код получившейся разметки представлен ниже:

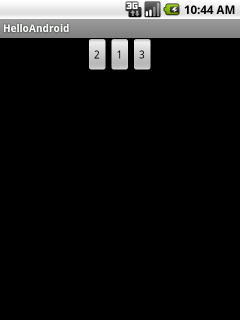
TableLayout. Эксперименты с gravity и layout\_weight

**RelativeLayout**

**RelativeLayout** - элементы этого способа разметки могут располагаться относительно друг друга, не следуя принципам поведения LinearLayout и TableLayout. Пример:

RelativeLayout. toLeftOf и toRightOf

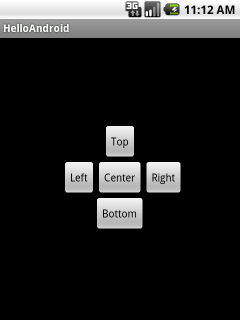
Первым элементом в списке размещения является кнопка с идентификатором btn1, однако фактически она будет второй слева. Это произойдет потому что слева от нее будет располагаться кнопка btn2, следуя инструкции **layout\_toLeftOf**="@id/btn1" (переводится как "слева от btn1"). А кнопка btn3 будет всегда располагаться справа от btn1, следуя инструкции **layout\_toRightOf**="@id/btn1" (переводится как "справа от btn1"). Горизонтальное расположение btn1 по центру достигается благодаря атрибуту layout\_centerHorizontal="true". Результат выполнения этого кода представлен на Рис. 10.

  
Рис 10. RelativeLayout. toLeftOf и toRightOf

Рассмотрим еще 2 атрибута, которые могут быть использованы в RelativeLayout: layout\_above и layout\_below. Атрибут **layout\_above** означает, что элемент будет располагаться над элементом, указанным в атрибуте. В свою очередь, атрибут**layout\_below** означает, что элемент будет располагаться под указанным в атрибуте элементом. Пример:

RelativeLayout. layout\_above и layout\_below

Результат выполнения этого кода представлен на Рис.11.

  
Рис 11. RelativeLayout. layout\_above и layout\_below