**Всё о LinearLayout - 1**

Этой статьёй, я открываю цикл с полным разбором функциональности Android виджетов и макетов (layout).

Начнём с макетов, а точнее разберём и перемоем все косточки диспетчеру компоновки [LinearLayout](http://developer.android.com/reference/android/widget/LinearLayout.html" \o "Посмотреть официальную документацию" \t "_blank).

В 2 словах, этот диспетчер позволяет компоновать внутри себя различные элементы один за одним либо вертикально, либо горизонтально. LinearLayout может быть корневым элементом макета (layout) и быть вложенным сам в себя сколько потребуется раз. Также можно взаимодействовать с другим компонвками, например, [RelativeLayout](http://developer.android.com/reference/android/widget/RelativeLayout.html" \o "Посмотреть официальную документацию" \t "_blank), как быть вложенным, так и быть *родителем.*

Начнём сразу с примеров или прямиком с места в карьер.

Я буду приводить XML код компоновки и тут же аналог кода на JAVA.

**Ориентация контента**

Для того, чтобы задать ориентацию, воспользуйтесь атрибутом orientationв xml коде макета или методом setOrientationв JAVA коде.

**В  xml**

android:orientation="horizontal"

android:orientation="vertical"

**В Java**

*// LinearLayout - элемент созданный в xml или java коде*

linearLayout.setOrientation(LinearLayout.HORIZONTAL);

linearLayout.setOrientation(LinearLayout.VERTICAL);

Хочу обратить ваше внимание, что по умолчанию LinearLayout использует горизонтальную ориентацию.

Рассмотрим для начала простые примеры.  
2 разметки одна с вертикальной, а другая с горизонтальной ориентацией и вложенные в неё 3 элемента.

**Вертикальная ориентация в XML**

<?xml version="1.0" encoding="utf-8"?>

<**LinearLayout** **xmlns:android**=**"http://schemas.android.com/apk/res/android"**

**android:orientation**=**"vertical"**

**android:gravity**=**"center\_horizontal"**

**android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"fill\_parent"**>

<**Button** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:text**=**"Кнопка 1"**/>

<**Button** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:text**=**"Кнопка 2"**/>

<**Button** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:text**=**"Кнопка 3"**/>

</**LinearLayout**>

**Вертикальная ориентация в JAVA**

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

LinearLayout linearLayout = **new** LinearLayout(**this**);

linearLayout.setOrientation(LinearLayout.VERTICAL);

linearLayout.setGravity(Gravity.CENTER\_HORIZONTAL);

ViewGroup.LayoutParams paramsLayout = **new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.FILL\_PARENT

);

linearLayout.setLayoutParams(paramsLayout);

ViewGroup.LayoutParams paramsButton = **new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

Button btn1 = **new** Button(**this**);

btn1.setText(**"Кнопка 1"**);

Button btn2 = **new** Button(**this**);

btn2.setText(**"Кнопка 2"**);

Button btn3 = **new** Button(**this**);

btn3.setText(**"Кнопка 3"**);

linearLayout.addView(btn1, paramsButton);

linearLayout.addView(btn2, paramsButton);

linearLayout.addView(btn3, paramsButton);

setContentView(linearLayout);

}

В результате обоих случаев выходит следующее:



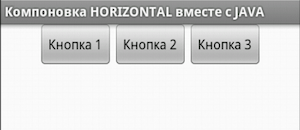
Для расположения кнопок в **горизонтальной разметке** нужно:  
в xml компоновке изменить атрибут android:orientation тега LinearLayout так:

android:orientation="horizontal"

а при использовании Java измените вызов setOrientation так:

linearLayout.setOrientation(LinearLayout.HORIZONTAL);

В итоге получим следующее:



С простыми примерами покончено.  
  
Теперь хотелось бы отдельно остановиться на вложенном классе LayoutParams и некоторых параметрах xml кода.

**Высота и ширина элементов**

Для того, чтобы система знала какую ширину и высоту задать элементу, у каждого элемента есть соответствующие параметры.  
В xml коде - это layout\_width и layout\_height. В JAVA для этого используется подкласс LayoutParams.  
Значения можно задавать, как в единицах измерения, например:

android:layout\_width="100dp"

android:layout\_height="40px"

*<!-- px использовать не рекомендуется -->*

*// java code*

elem.setLayoutParams(**new** LinearLayout.LayoutParams(100, 40));

, так и определёнными значениями:

| **Назначение** | **XML значение** | **JAVA значение** |
| --- | --- | --- |
| Всё свободное пространство в заданном направлении | fill\_parent | ViewGroup.LayoutParams.FILL\_PARENT |
| Всё свободное пространство в заданном направлении (работает начиная с версии 2.2) | match\_parent | ViewGroup.LayoutParams.MATCH\_PARENT |
| Элемент занимает ровно столько, сколько ему нужно, но не больше, чем позволяет свободное пространство | wrap\_content | ViewGroup.LayoutParams.WRAP\_CONTENT |

Таким образом, если мы хотим использовать компоновку или элемент представления во всю ширину и только в требующуюся высоту, код будет таким:

**xml:**

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

**java:**

view.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

Кстати, вы заметили, что в таблице у нас получилось 2 разных значения с одинаковым назначением?  
Да? Так давайте же разберёмся в их отличии.

**Сравнение FILL\_PARENT И MATCH\_PARENT**

Константа FILL\_PARENT объявлена в Android 2.2 устаревшей и заменена константой MATCH\_PARENT. Но  на самом деле -  это просто замена одного имени другим и даже значение осталось равным -1.  
  
**Какое значение использовать?**  
Можно использовать в зависимости от того, какой API-интерфейс Android необходимо использовать в приложении, либо построить приложение для версии Android предшествующей версии  Android 2.2, и полагаться на прямую совместимость, либо создать приложение для версии 2.2 или выше и установить minSdkVersion в наименьшую версию Android.

Теперь давайте применим полученные навыки и создадим макет с использованием обеих ориентаций чуточку посложнее.

**xml - компоновка**

<?xml version="1.0" encoding="utf-8"?>

<**LinearLayout** **xmlns:android**=**"http://schemas.android.com/apk/res/android"**

**android:orientation**=**"vertical"**

**android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"fill\_parent"**

>

<**LinearLayout** **android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"wrap\_content"**

**android:orientation**=**"horizontal"**

**android:paddingTop**=**"5dp"**

**android:paddingBottom**=**"5dp"**

**android:background**=**"#ddd"**

>

<**ImageView** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:layout\_gravity**=**"center"**

**android:layout\_marginRight**=**"10dp"**

**android:layout\_marginLeft**=**"5dp"**

**android:scaleType**=**"fitCenter"**

**android:src**=**"@drawable/anonym"**

/>

<**LinearLayout** **android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"wrap\_content"**

**android:orientation**=**"vertical"**>

<**TextView** **android:layout\_width**=**"wrap\_content"**

**android:layout\_height**=**"wrap\_content"**

**android:text**=**"Аноним"**

**android:textSize**=**"20sp"**

**android:textStyle**=**"bold"**

/>

<**LinearLayout** **android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"wrap\_content"**

**android:orientation**=**"horizontal"**

>

<**TextView** **android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"wrap\_content"**

**android:text**=**"Пропущенных вызовов от абонента - 1, время 20:48 09/12"**

**android:layout\_weight**=**"1.0"**

**android:textSize**=**"13sp"**

**android:maxLines**=**"2"**/>

<**TextView** **android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"wrap\_content"**

**android:layout\_weight**=**"2.0"**

**android:layout\_gravity**=**"bottom"**

**android:gravity**=**"right"**

**android:paddingRight**=**"5dp"**

**android:text**=**"11/11/12"**

**android:textSize**=**"12sp"**

**android:textColor**=**"#00f"**

/>

</**LinearLayout**>

</**LinearLayout**>

</**LinearLayout**>

<**Button** **android:layout\_width**=**"fill\_parent"**

**android:layout\_height**=**"wrap\_content"**

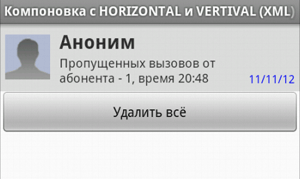
**android:text**=**"Удалить всё"**

**android:gravity**=**"center"**

/>

</**LinearLayout**>

Получилось следующее:



Разберём код.

1. Вначале мы создали контейнер в вертикальной ориентации и положили ему внутрь 2 элемента - 2-ой контейнер LinearLayout и кнопку Button.
2. Второму контейнеру LinearLayout присвоим горизонтальную ориентацию, серый фоновый цвет и отступы сверху и снизу в 5dp. Зададим ширину во всё свободное пространство, а высоту в требуемое. Далее вложим внутрь ещё 2 элемента – изображение ImageView и 3-ий контейнер LinearLayout.
3. На этот раз выставим ориентацию в вертикальное положение и зададим ширину во всё свободное пространство, а высоту в требуемое и добавим внутрь текст TextView и 4-ый контейнер LinearLayout.
4. 4-ый контейнер сделаем в горизонтальной ориентации, зададим ему ширину во всё свободное пространство, а высоту в требуемое и добавим внутрь 2 элемента TextView. Элементам назначим вес layout\_weight, но об этом атрибуте мы поговорим в следующей статье.

В итоге у нас получилась такая схема:

* LinearLayout – vertical
  + LinearLayout – horizontal
    - ImageView
    - LinearLayout – vertical
      * TextView
      * LinearLayout – horizontal
        + TextView
        + TextView
  + Button

Ниже сделаем тоже самое, но используя только JAVA.

**Тоже самое, но на JAVA**

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

ImageView contactImage = **new** ImageView(**this**);

LinearLayout.LayoutParams paramContactImage = **new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

paramContactImage.leftMargin = 5;

paramContactImage.rightMargin = 10;

paramContactImage.gravity = Gravity.CENTER;

contactImage.setLayoutParams(paramContactImage);

contactImage.setImageResource(R.drawable.anonym);

contactImage.setScaleType(ImageView.ScaleType.FIT\_CENTER);

TextView contactName = **new** TextView(**this**);

contactName.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.WRAP\_CONTENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

contactName.setText(**"Аноним"**);

contactName.setTextSize(20);

contactName.setTypeface(**null**, Typeface.BOLD);

TextView messageText = **new** TextView(**this**);

LinearLayout.LayoutParams paramMessage = **new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

paramMessage.weight = (**float**) 1.0;

messageText.setLayoutParams(paramMessage);

messageText.setText(**"Пропущенных вызовов от абонента - 1, время 20:48 09/12"**);

messageText.setTextSize(13);

messageText.setMaxLines(2);

TextView dateText = **new** TextView(**this**);

LinearLayout.LayoutParams paramDate = **new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

);

paramDate.weight = (**float**) 2.0;

paramDate.gravity = Gravity.BOTTOM;

dateText.setLayoutParams(paramDate);

dateText.setText(**"11/11/12"**);

dateText.setGravity(Gravity.RIGHT);

dateText.setPadding(0, 0, 5, 0);

dateText.setTextSize(12);

dateText.setTextColor(Color.parseColor(**"#0000ff"**));

Button btnDelete = **new** Button(**this**);

btnDelete.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

btnDelete.setText(**"Удалить всё"**);

btnDelete.setGravity(Gravity.CENTER);

LinearLayout messagePanel = **new** LinearLayout(**this**);

messagePanel.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

messagePanel.setOrientation(LinearLayout.HORIZONTAL);

messagePanel.addView(messageText);

messagePanel.addView(dateText);

LinearLayout contactPanel = **new** LinearLayout(**this**);

contactPanel.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

contactPanel.setOrientation(LinearLayout.VERTICAL);

contactPanel.addView(contactName);

contactPanel.addView(messagePanel);

LinearLayout item = **new** LinearLayout(**this**);

item.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.WRAP\_CONTENT

));

item.setOrientation(LinearLayout.HORIZONTAL);

item.setBackgroundColor(Color.parseColor(**"#dddddd"**));

**int** dp = (**int**) (getResources().getDisplayMetrics().density \* 5);

item.setPadding(0, dp, 0, dp);

item.addView(contactImage);

item.addView(contactPanel);

LinearLayout root = **new** LinearLayout(**this**);

root.setLayoutParams(**new** LinearLayout.LayoutParams(

ViewGroup.LayoutParams.FILL\_PARENT,

ViewGroup.LayoutParams.FILL\_PARENT

));

root.setOrientation(LinearLayout.VERTICAL);

root.addView(item);

root.addView(btnDelete);

setContentView(root);

}

На сегодня достаточно.  
В следующий раз я продолжу разговор про эту компоновку и расскажу про другие атрибуты.  
  
Код данного поста можно скачать/посмотреть на гитхабе - <https://github.com/devpad/Android-LinearLayout-1>