

레이아웃(Layout)

레이아웃에서 사용되는 공통 속성

- ✓ **orientation** : 레이아웃 안에 배치할 위젯의 수직 또는 수평 방향을 설정
- ✓ **gravity** : 위젯안의 내용물의 정렬 방향 설정
- ✓ **layout_gravity** : 레이아웃 안의 위젯의 정렬 방향 설정
- ✓ **padding** : 레이아웃 안에 배치할 위젯의 여백을 설정
- ✓ **layout_weight** : 레이아웃이 전체 화면에서 차지하는 공간의 가중 값을 설정
- ✓ **baselineAligned** : 레이아웃 안에 배치할 위젯들을 보기 좋게 정렬

리니어 레이아웃(LinearLayout)

➤ **ViewGroup**으로부터 파생되는 클래스.

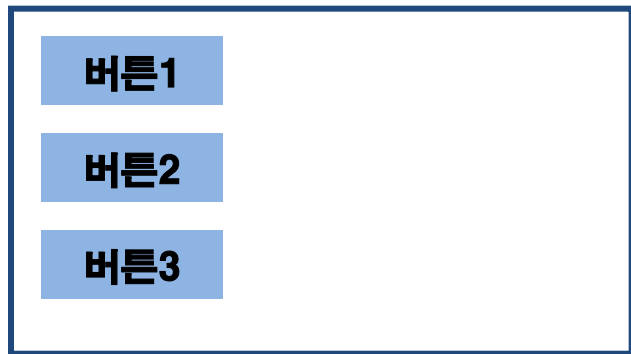
- ✓ 차일드 뷰를 수평, 수직으로 일렬 배치하는 레이아웃으로, 가장 단순하면서 직관적이며 사용빈도 높음
- ✓ 왼쪽 위부터 아래쪽 또는 오른쪽으로 차례로 배치

➤ **orientation**

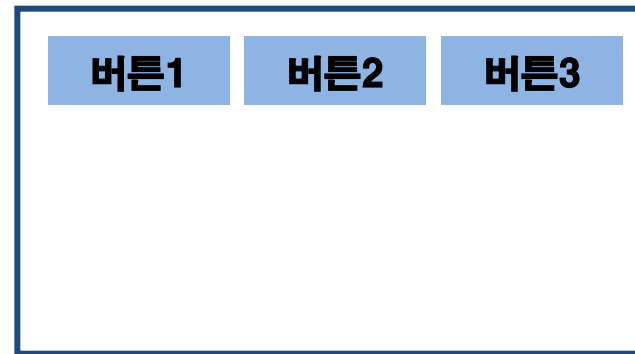
뷰의 배치 방향을 결정하는 속성. (디폴트는 **horizontal**)

vertical : 차일드를 위에서 아래로 수직으로 배열

horizontal : 차일드를 왼쪽에서 오른쪽으로 수평 배열



[vertical]



[horizontal]

리니어 레이아웃 - orientation

```
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android  
    android:orientation="horizontal"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >
```



[Horizontal]

리니어 레이아웃 - gravity

내용물의 위치를 지정하며, 수평, 수직 방향에 대해 각각 정렬 방식을 지정 가능
두 속성을 같이 지정할 때는 “|” 연산자를 이용하며, 이 때 연산자 양쪽으로 공백이 전혀
없어야 함

상수	설명
center_horizontal	수평으로 중앙에 배치한다.
left	컨테이너의 왼쪽에 배치하며, 크기는 바뀌지 않는다.
right	컨테이너의 오른쪽에 배치한다.
fill_horizontal	수평 방향으로 가득 채운다.
center_vertical	수직으로 중앙에 배치한다.
top	컨테이너의 상단에 배치하며, 크기는 바뀌지 않는다.
bottom	컨테이너의 하단에 배치한다.
fill_vertical	수직 방향으로 가득 채운다.
center	수평으로나 수직으로 중앙에 배치한다.
fill	컨테이너에 가득 채우도록 수직, 수평 크기를 확장한다.

리니어 레이아웃 – layout_gravity

gravity 와 동일 속성이 적용되지만 대상이 다르다.

gravity 는 **view** 의 내용물을 어디에 배치 할 것 인가를 지정.

layout_gravity 는 부모의 어디다 **view** 를 배치 할 것 인가의 문제

리니어 레이아웃 – layout_weight

weight에 따라 부모 영역의 여유 여백을 자식 뷰가 차지

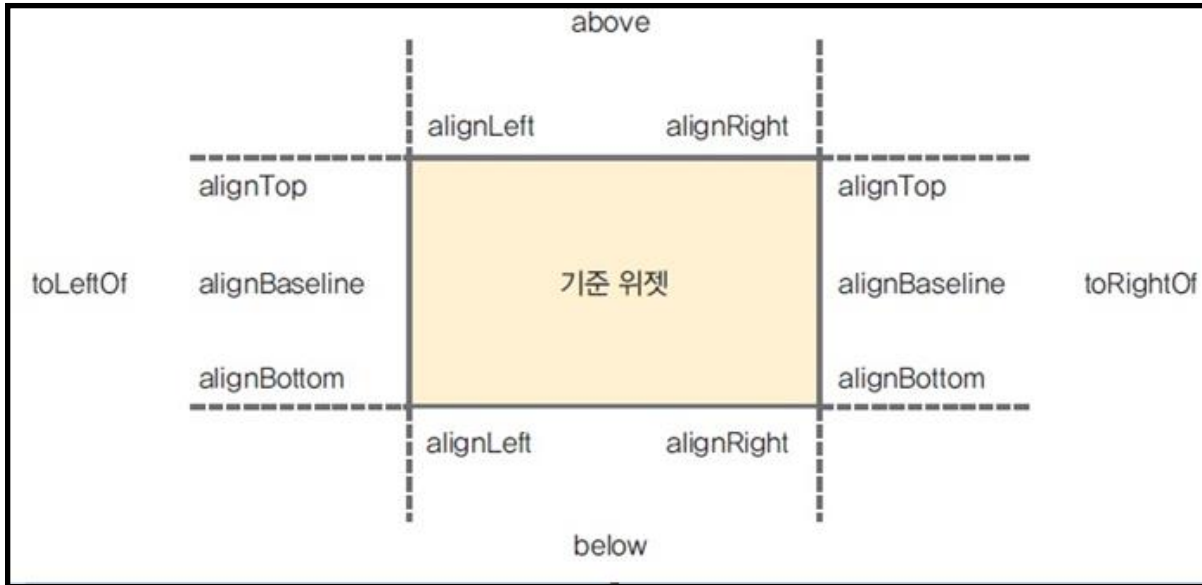
weight가 0이면 자신의 고유한 크기만큼, 1 이상이면 형제 뷰와의 비율에 따라 부모의 영역을 균등하게 배분

상대 레이아웃(RelativeLayout)

위젯과 부모와의 위치 관계 또는 위젯끼리의 관계를 지정함으로써 뷰를 배치
위젯끼리의 관계 지정에 위하여 기준이 되는 위젯에 **id**를 반드시 지정해야 함

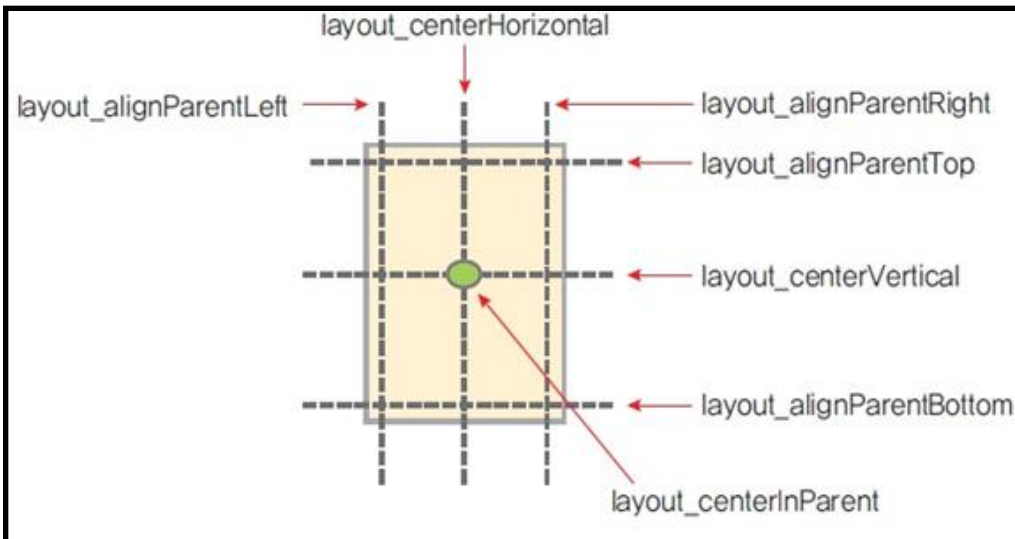
속성	설명
layout_above	~의 위에 배치한다
layout_below	~의 아래에 배치한다
layout_toLeftOf	~의 왼쪽에 배치한다
layout_toRightOf	~의 오른쪽에 배치한다
layout_alignLeft	~의 왼쪽 변을 맞춘다
layout_alignRight	~의 오른쪽 변을 맞춘다
layout_alignTop	~의 위쪽 변을 맞춘다
layout_alignBottom	~의 아래쪽 변을 맞춘다
layout_alignBaseline	~와 베이스라인을 맞춘다

상대 레이아웃(RelativeLayout)



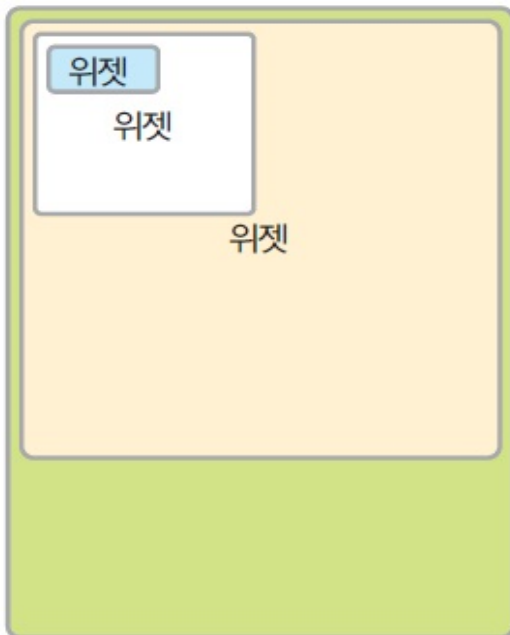
상대 레이아웃(RelativeLayout)

속성	설명
layout_alignParentLeft	true이면 부모와 왼쪽 변을 맞춘다
layout_alignParentTop	true이면 부모와 위쪽 변을 맞춘다
layout_alignParentRight	true면 부모와 오른쪽 변을 맞춘다
layout_alignParentBottom	true면 부모와 아래쪽 변을 맞춘다
layout_centerHorizontal	true이면 부모의 수평 중앙에 배치한다
layout_centerVertical	true이면 부모의 수직 중앙에 배치한다
layout_centerInParent	true이면 부모의 수평, 수직 중앙에 배치한다



프레임 레이아웃(FrameLayout)

- 차일드를 배치하는 규칙 없이 모든 차일드는 프레임의 좌상단에 나타나며, 차일드가 두 개 이상일 때는 추가된 순서대로 겹쳐서 표시
- **ViewGroup**의 서브 클래스로 **addView**, **removeView** 등의 메서드로 실행 중에 차일드를 추가, 삭제할 수 있으며, **getChildCount** 메서드로 차일드의 개수를 조사 함
- 차일드의 보임 상태는 개별 뷰의 **visibility** 속성을 사용하여 조정하며 실행 중에도 조건에 따라 뷰의 보임 상태를 변경할 수 있다.



그리드 레이아웃(GridLayout)

GridLayout 은 4.0 에 새로 추가된 레이아웃이다.

가로, 세로의 셀로 나누어 격자 모양의 표에 차일드를 배치한다.

주로 사각형을 반듯하게 뷰로 배치할때 사용되지만 셀의 위치와 크기를 다양하게 변경 가능하며 셀끼리 병합도 가능하다.

TableLayout 과 비슷하지만 차일드를 순서대로 배치한다는 점에서 **LinearLayout** 과도 비슷하다.

TableLayout과 **LinearLayout** 을 섞어 놓은 모양새인데 리니어처럼 동작하되 일정 개수를 넘어서면 자동으로 개행되어 결국 테이블과 유사한 배치를 만든다.

그리드 레이아웃 속성

GridLayout 자체 태그에서 사용하는 속성

orientation	배치의 방향 지정
columnCount	최대 열의 개수 지정. 한행이 이 개수를 초과하면 아래행으로 자동 개행
rowCount	최대 행 개수를 지정. 한열이 이 개수를 초과하면 오른쪽 열로 자동 개행

그리드 레이아웃 안에 포함될 위젯에서 사용하는 속성

useDefaultMargins	차일드 뷰의 레이아웃에 별다른 지정이 없으면 차일드의 속성을 참조하여 계산한 디폴드 마진을 사용한다. 이 값이 false 이면 마진은 0 으로 처리
layout_column	차일드가 배치될 셀의 열 좌표지정.
layout_row	차일드가 배치될 셀의 행 좌표 지정
layout_columnSpan	차일드가 차지할 열 수를 지정
layout_rowSpan	차일드가 차지할 행 수를 지정
layout_gravity	fill, fill_vertical, fill_horizontal layout_rowSpan 또는 layout_columnSpan 으로 확장되었을 때 위젯을 확장된 셀에 채우는 효과가 발생

그리드의 배치 방향

- **orientation** 속성을 이용하여 차일드의 방향 지정. 기본값은 **horizontal**
- **horizontal** 에서의 **rowCount** 는 의미 없다.
- **vertical** 에서의 **columnCount** 는 의미 없다.

```
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="3"
    android:rowCount="3"
    android:orientation="horizontal">

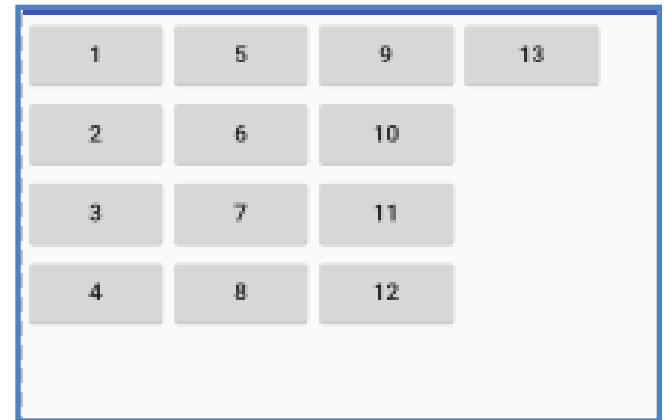
    <Button android:text="1" />
    <Button android:text="2" />
    .....
    <Button android:text="12" />
    <Button android:text="13" />
</GridLayout>
```



그리드의 배치 방향

➤ vertical 에서의 위젯이 붙는 방향

```
<GridLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:rowCount="4"  
    android:orientation="vertical">  
  
    <Button android:text="1" />  
    <Button android:text="2" />  
    .....  
    <Button android:text="12" />  
    <Button android:text="13" />  
</GridLayout>
```



그리드의 배치 방향

- **columnCount, rowCount**를 지정하지 않으면 한줄로 나열

```
<GridLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="vertical">  
  
    <Button android:text="1" />  
    <Button android:text="2" />  
    .....  
    <Button android:text="12" />  
    <Button android:text="13" />  
</GridLayout>
```



그리드 셀의 크기

GridLayout 의 차일드는 셀이라는 한정된 공간에 배치되므로 **layout_width**, **layout_height** 속성을 따로 지정하지 않는다. 셀 내의 차일드의 크기는 항상 **wrap_content** 이며 차일드의 고유 크기만큼만 차지한다. **match_parent** 지정해도 무시한다.

<GridLayout

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:columnCount="3"
```

>

```
    <Button android:text="1"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

```
    <Button android:text="2" />
```

```
    <Button android:text="3" />
```

```
    <Button android:text="4" />
```

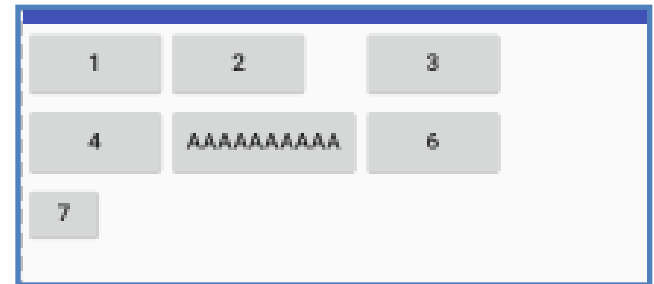
```
    <Button android:text="AAAAAAAAAA" />
```

```
    <Button android:text="6" />
```

```
    <Button android:text="7"
```

```
        android:layout_width="50dp"
        android:layout_height="40dp" />
```

</GridLayout>



셀의 크기

셀 내에서의 기본 크기는 `wrap_content` 인데 다른 셀의 `content` 가 길어서 영역이 커진 경우 셀 영역을 다 차지하게 하기 위해서는 `layout_width` 속성이 아닌 `layout_gravity` 속성을 이용한다.

<GridLayout

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:columnCount="3">
```

```
    <Button android:text="1" />
```

```
    <Button android:text="2" android:layout_gravity="fill_horizontal" />
```

```
    <Button android:text="3" />
```

```
    <Button android:text="4" />
```

```
    <Button android:text="AAAAAAAAAAAA" />
```

```
    <Button android:text="6" />
```

```
    <Button android:text="7" />
```

</GridLayout>



셀의 좌표

`layout_row`, `layout_column` 속성을 이용하여 셀의 위치를 지정할수 있다.

하나의 차일드의 좌표를 지정하면 그 이후 차일드는 다시 규칙대로 자리를 잡는다.

```
<GridLayout
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:columnCount="3"
```

```
>
```

```
    <Button android:text="1" />
```

```
    <Button android:text="2" />
```

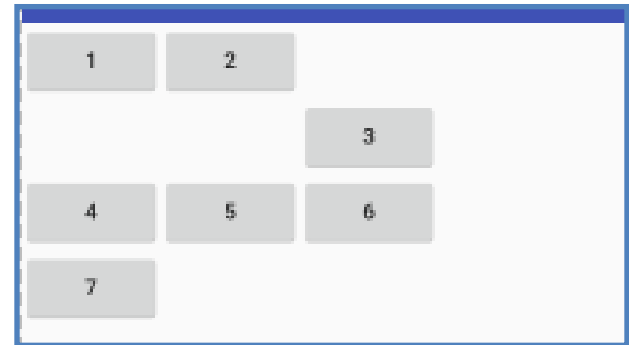
```
    <Button android:text="3" android:layout_row="1"
```

```
        android:layout_column="2" />
```

```
    .....
```

```
    <Button android:text="7" />
```

```
</GridLayout>
```



셀의 좌표

하나의 셀이 길때 차일드를 한 셀안에 두개 위치시킬수도 있다.

물론 이때 겹치지 않게 하기 위해 `layout_gravity` 속성을 이용한다.

<GridLayout

`android:layout_width="wrap_content"`

`android:layout_height="wrap_content"`

`android:columnCount="3">`

`<Button android:text="1 - AAAAAAAAAABBBBBBBB" />`

.....

`<Button android:text="2" />`

`<Button android:text="5"`

`android:layout_row="1"`

`android:layout_column="0"`

`android:layout_gravity="right"/>`

.....

`<Button android:text="7" />`

`</GridLayout>`



셀 병합

하나의 셀이 하나의 차일드를 배치하는것이 원칙이지만 셀을 병합하여 여러 셀에 하나의 차일드를 위치시키는것도 가능하다.

`layout_columnSpan`, `layout_rowSpan` 속성을 이용한다.

```
<GridLayout
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:columnCount="3">
```

```
    <Button android:text="1" />
```

```
    <Button android:text="2" android:layout_rowSpan="2" />
```

```
    <Button android:text="3" />
```

```
    <Button android:text="4" />
```

```
    <Button android:text="5" />
```

```
</GridLayout>
```



셀 병합

하나의 셀이 하나의 차일드를 배치하는것이 원칙이지만 셀을 병합하여 여러 셀에 하나의 차일드를 위치시키는것도 가능하다.

`layout_columnSpan`, `layout_rowSpan` 속성을 이용한다.

<GridLayout

`android:layout_width="wrap_content"`

`android:layout_height="wrap_content"`

`android:columnCount="3">`

`<Button android:text="1" />`

`<Button android:text="2" android:layout_rowSpan="2"`

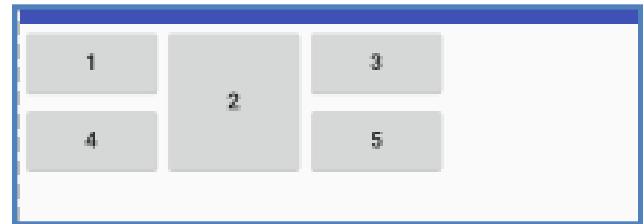
`android:layout_gravity="fill" android:gravity="center" />`

`<Button android:text="3" />`

`<Button android:text="4" />`

`<Button android:text="5" />`

`</GridLayout>`



셀 병합

<GridLayout

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:columnCount="3">

<Button android:text="1" />

<Button android:text="2"

android:layout_rowSpan="2"

android:layout_columnSpan="2"

android:layout_gravity="fill" android:gravity="center" />

<Button android:text="3" />

<Button android:text="4" />

<Button android:text="5" />

</GridLayout>

