Android ve Tasarım #1 : Etkili Layout Seçimi ve Kullanımı

Muhammed Burak Çakır Follow
Aug 14, 2019 · 7 min read

Bildiğiniz gibi **tasarımlar** programladığımız Android uygulamalarda büyük bir öneme sahip. Kullanıcı gözüyle uygulamalara baktığımızda, insanların tasarıma ve uygulamanın kendi içindeki akışına önem verdiğini görebiliriz. Bu sebeple, yazımda Layout kavramını ve Layout tasarımını giriş seviyesinde sizlere aktarmak istiyorum. Kullanımlarını ben de zamanla oturtmaya çalışıyorum. Bu yüzden hatalarım olacaktır fakat genel çerçevesiyle Layout mantığını kavratacak bir yazı olduğunu düşünüyorum. Yazıma sıkça rastlayacağınız "**component**" kavramıyla başlamak isterim.

Component Nedir?

Componentler basitçe, **View** tarafında kullandığımız nesnelerdir. Tasarımımızı componentler ile oluştururuz. Componentlere için "**Widget**" kavramı da kullanılmakta

fakat ben bu yazı için component kavramını kullanacağım.

Ör: TextView, ImageView, EditText…

Layout Nedir?

Android projelerimizde, Projemizin View yani arayüz kısmını .xml formatlı olarak tasarlarız. Bu arayüzler, layout dosyaları aracılığıyla oluşturulur.

Layout Klasörü

Layout, yaptığımız tasarımda View componentlerin yaşama alanıdır diyebiliriz. Tasarımlarda kalıp görevi görerek sağladığı özelliklerine göre içinde kullanılan componentlere düzen sağlar, nasıl gözükmeleri gerektiğini belirler.

Android'de farklı layoutlar vardır ve farklı amaçlar için kullanılır. Android programlamaya başlandığında sık rastlanan hizalama ve boşluk hatalarını kolayca çözmemizi sağlayan ve layoutlarda kullanmamız gereken 2 temel tasarım özelliğini göstermek istiyorum.

- **wrap_context** : İçerisinde bulunan componentler kadar yer kaplar.

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

kodlarıyla xml dosyamızdan ayarlayabilirken,

Android Studio — Attributes

Attributes kısmından da ayarlanabilmektedir.

- **match_parent** : İçinde bulunduğu alanı tüm doğrultusuyla kapsar.

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

Şimdi Layout türlerine giriş yapabiliriz.

1) LinearLayout

Android programlamaya başlayanların, ilk zamanlarda sık sık denk geldiği layoutlardan biridir.

Linear Layout, temel olarak içinde bulunan componentleri yatay (horizontal) veya dikey (vertical) olarak **sırayla** hizalamaya yarar.

Bunu belirlemek için xml özelliklerimizden veya daha önce söylediğimiz “Attributes” kısmından kullanılan Layout’un orientation(yön) ayarlamasını yapabiliriz.

Yatay Hizalama:

LinearLayout (horizontal)

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
4  xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
5   xmlns:tools="http://schemas.android.com/tools"
6   android:layout_width="match_parent"
7   android:layout_height="match_parent"
8   android:gravity="center"
9   android:orientation="horizontal"
10  tools:context=".MainActivity">
11
12  <Button
13      android:id="@+id/button1"
14      android:layout_width="wrap_content"
15      android:layout_height="wrap_content"
16      android:layout_weight="1"
17      android:text="YATAY-1" />
18
19  <Button
20
21      android:id="@+id/button2"
22      android:layout_width="wrap_content"
23      android:layout_height="wrap_content"
24      android:layout_weight="1"
25      android:text="YATAY-2" />
26
27  <Button
28
29      android:id="@+id/button3"
30      android:layout_width="wrap_content"
31      android:layout_height="wrap_content"
32      android:layout_weight="1"
33      android:text="YATAY-3" />
34
35  </LinearLayout>
```

linearLavout horizontal.xml hosted with ♡ by GitHub view raw

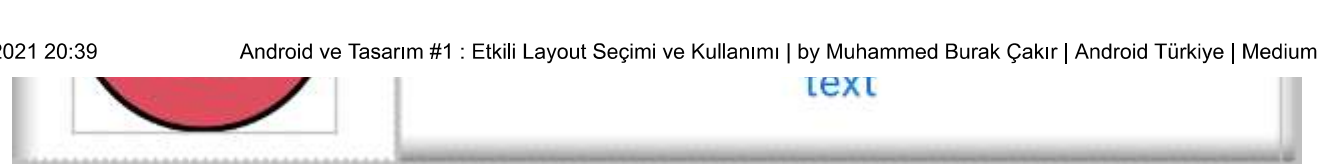
Dikey Hizalama:

LinearLayout (vertical)

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <Button

        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DİKEY-1" />

    <Button

        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DİKEY-2" />

    <Button

        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="DİKEY-3" />

</LinearLayout>
```

linearLayout vertical.xml hosted with ♡ by GitHub view raw

Şuan ki paylaşacağım örnek ise bir önceki projemde yaptığım sadece LinearLayout kullanarak tasarlamış olduğum basit bir CardView:



LinearLayoutlar ile CardView Tasarımı

LinearLayoutlar ile yaptığımız tasarımlarda hizalamayı doğru yaparsak, uygulamamız ileride farklı boyutlardaki cihazlarda da sıkıntısız olarak çalışacaktır. Çünkü orantıyı kurmak, elle sayı girmekten her zaman daha iyidir. Sayılar değişse bile program önceden kurgulanan orantıya bağlı kalarak tasarım düzenini koruyacaktır.

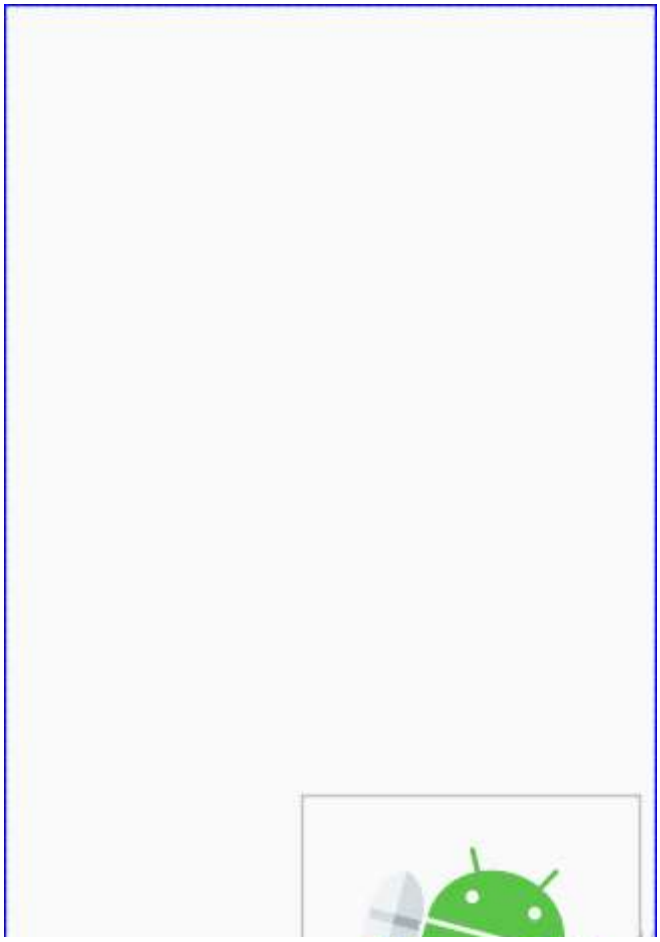
2) RelativeLayout

Relative Layout, temelinde programcıya tasarımda özgürlük tanır. Componentleri istediğimiz şekilde hizalayabilir ve sürükleyebiliriz. LinearLayout'tan farkı daha özgür olması ve farklı özellikler içermesidir. İçindeki componentleri, etrafındaki bileşenlere göre sağa, sola, yukarı veya aşağı hizalayabiliriz.

Bileşenleri, ekran içerisindeki bir kenara veya köşeye hizalamak için,

- **layout_alignParent** özelliğini kullanabiliriz.

Örnek olarak, layout_alignParentBottom ile alt kenara, layout_alignParentRight ile sağ kenara hizalayalım.



alignParent ile Hizalama

Kenara sıkışması hoş gözükmezdi. Bu yüzden 10dp margin ayarı yaptım.

Margin anlam olarak “kenar boşluğu” demektir ve Android’de tasarım elementlerimizin birbirinden ayrılması için önem taşır.

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

<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView

        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_margin="10dp"
        app:srcCompat="@drawable/android" />

</RelativeLayout>
```

relativeLayout_alignParent.xml hosted with ♡ by GitHub view raw

Görselimizi layout kenarlarına göre hizaladık. Peki etrafta başka componentler olursa, tasarımımız daha karmaşık bir hale gelirse ne yapacağız?

Bunun için kullanacağımız farklı hizalama yöntemleri var.

1- Dışarıdan Hizalama

- **layout_above** : seçilen componenti ID’si verilen diğer componentin **üstünden başlayarak dışarıya doğru** hizalar.

- **layout_below** : seçilen componenti ID'si verilen diğer componentin **altından başlayarak dışarıya doğru** hizalar.
- **layout_toRightOf** : seçilen componenti ID'si verilen diğer componentin **sağından başlayarak dışarıya doğru** hizalar.
- **layout_toLeftOf** : seçilen componenti ID'si verilen diğer componentin **solundan başlayarak dışarıya doğru** hizalar.

2- İçeriden Hizalama

- **layout_alignLeft** : seçilen component'i ID'si verilen diğer componentin **solundan başlayarak içine doğru** hizalar.
- **layout_alignRight** : seçilen component'i ID'si verilen diğer componentin **sağından başlayarak içine doğru** hizalar.
- **layout_alignBottom** : seçilen component'i ID'si verilen diğer componentin **altından başlayarak içine doğru** hizalar.
- **layout_alignTop** : seçilen component'i ID'si verilen diğer componentin **üstünden başlayarak içine doğru** hizalar.

Tanımlar çok yakın değil mi ? Okuduğunuzda kafanızda karışabilir. Karmaşıklığı çözelim ve kavramları anlamaya çalışalım.

Dışarıdan hizalamada, birbirine göre hizalanması gereken 2 component iç içe çakışmaz. Above(üst) veya Below(alt)'dan başlayarak o componentin dışına doğru hizalar.

layout_alignTop ve layout_above özelliklerini karşılaştıracağımız basit bir örnekle görelim.

layout_above:

layout_above kullanımı

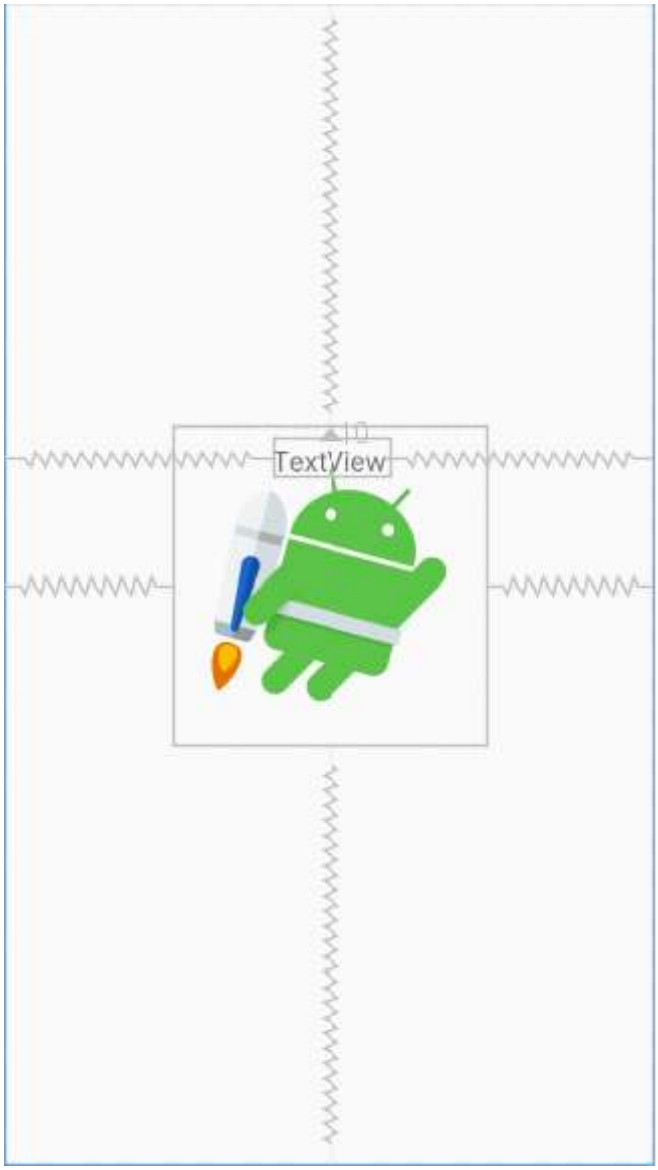
```xml
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView

        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_margin="10dp"
        app:srcCompat="@drawable/android" />

    <TextView

        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@id/imageView"
        android:layout_centerHorizontal="true"
```

```
27          android:text="TextView"
28          android:textSize="18sp" />
29
30  </RelativeLayout>
```

relativeLayout_above.xml hosted with ♡ by GitHub view raw

Burada TextView componentinde layout_above olarak kullandığımız ImageView'in ID'sini verdik ve artık TextView, ImageView'in üst sınırından başlayarak görselin dışarısına doğru konum almıştır. Farkın görülebilmesi açısından 10dp'lik küçük bir margin ayarı verdim.

layout_alignTop:



layout_alignTop kullanımı

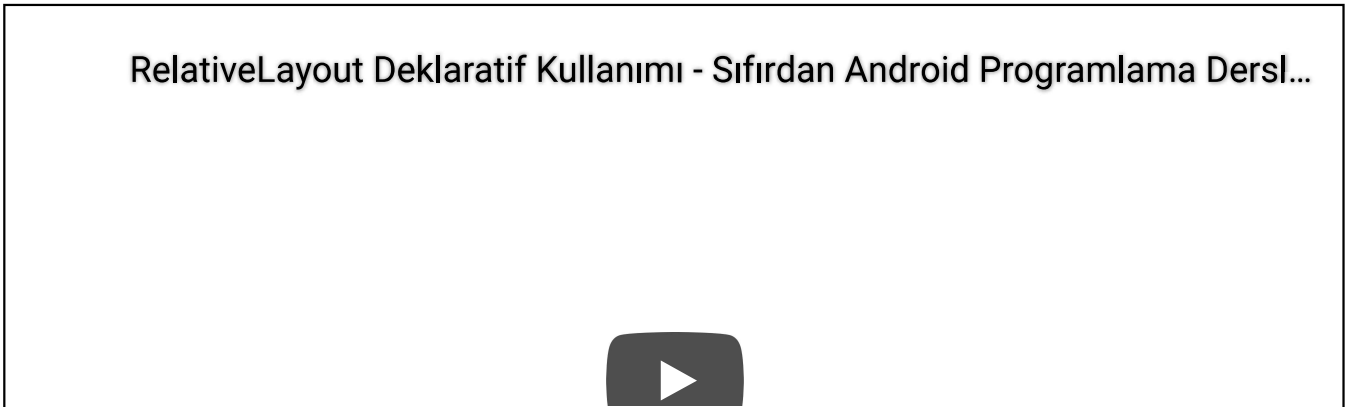
```
1   <?xml version="1.0" encoding="utf-8"?>
2
3   <RelativeLayout
```

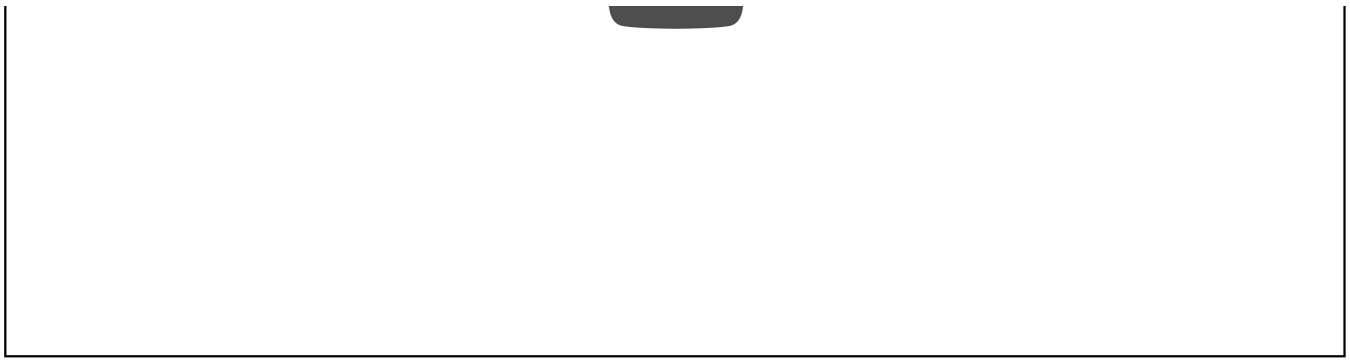
```
4
5       xmlns:android="http://schemas.android.com/apk/res/android"
6       xmlns:app="http://schemas.android.com/apk/res-auto"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent">
9
10      <ImageView
11
12          android:id="@+id/imageView"
13          android:layout_width="200dp"
14          android:layout_height="200dp"
15          android:layout_centerHorizontal="true"
16          android:layout_centerVertical="true"
17          android:layout_margin="10dp"
18          app:srcCompat="@drawable/android" />
19
20      <TextView
21
22          android:id="@+id/textView3"
23          android:layout_width="wrap_content"
24          android:layout_height="wrap_content"
25          android:layout_alignTop="@id/imageView"
26          android:layout_centerHorizontal="true"
27          android:text="TextView"
28          android:textSize="18sp" />
29
30  </RelativeLayout
```

relativeLayout_alignTop.xml hosted with ♡ by GitHub view raw

Burada ise, TextView componentinin alignTop özelliğine ImageView verdik. Yukarıdaki örneğe benzer olarak yerleşime yine görselin yukarısından başladı. Fakat, görselin içine doğru yerleştiğinden görsel ve yazı çakışmıştır. Aynı şekilde farkın görülebilmesi açısından 10dp'lik küçük bir margin ayarı verdim.

Ayrıca, Uygula Öğren kanalının videosunu sizlere tavsiye ediyorum.





3) GridLayout

GridLayout, TableLayout'un daha gelişmiş halidir. TableLayout veri gösterimi için kullanılabilir fakat cansızdır ve dinamik olarak kullanışsızdır. Daha gelişmiş özellikleri içeren bir tablo görünümü için GridLayout kullanmalıyız. Çünkü GridLayout aynı zamanda ListView'e yakın mantıkta çalışır. Adapter ile kullanılabilir, gezinme, sürükleme vb. imkanlar tanır. Sütun veya satırlara istediğimiz componentleri yerleştirip onları düzene sokmamıza olanak sağlar. Bazı temel özelliklerini incelemeye başlayalım.

Satır ve sütün sayılarını kendimiz belirleriz.

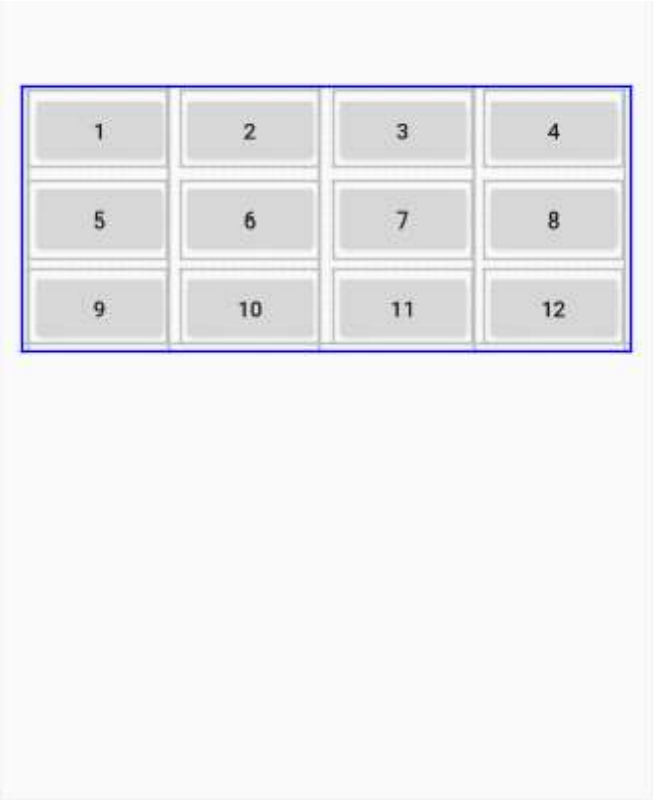
Bunun için

- **columnCount**
- **rowCount** özelliklerini kullanırız.

```
1  <GridLayout
2  android:columnCount="4"
3  android:rowCount="3"
4  </GridLayout>
```

gridLayout_count.xml hosted with ♡ by GitHub view raw

Kodunu xml dosyamıza eklediğimizde, 4x3'lük bir tablomuz oluşacaktır.



GridLayout Tablo Oluşturma

Peki, içine component eklediğimizde yerlerine nasıl karar vereceğiz?

Mesela 7 ve 8 sayılarının yerlerini değiştirebilir miyiz?

Evet. Bunun için

- **layout_row**
- **layout_column** özelliklerine indeks verebiliriz.

Örnek olarak:

```
<Button

    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_row="1"
    android:layout_column="3"
    android:text="7" />

<Button

    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:layout_row="1"
```

```
14      android:layout_column="2"
15      android:text="8" />
```

gridLayout_row_column.xml hosted with ♡ by GitHub view raw

GridLayout row ve column kullanımı

ile istediğimiz bileşenin GridLayout'ta ki yerlerini x ve y koordinatları olarak düşünerek değiştirebiliriz.

Unutmamız gereken detay şudur: satır ve sütunlarda **ilk indis 0'dır.**

Böylece Android bize bileşenlerin yerlerini row ve column ayarları vererek değiştirme imkanı tanıyor.

Ayrıca,

```
android:useDefaultMargins="true"
```

komutu ile layoutun varsayılan kenar boşluklarını kullanabilmesini sağlar.

Peki, her satırın veya sütunun kullanması gerektiği alanı değiştirmemiz gerektiğinde ne yapacağız?

Bunun için

- **layout_rowSpan**
- **layout_columnSpan** özelliklerini kullanabilirirz.

```
<Button

    android:layout_columnSpan="2"
    android:layout_rowSpan="2"
    android:layout_gravity="fill"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="3" />
```

button_span.xml hosted with ♡ by GitHub view raw

İle butonumuza 2x2'lik alan tanıyabiliriz.

rowSpan ve columnSpan kullandığımızda, doldurmanın başarılı olması için,

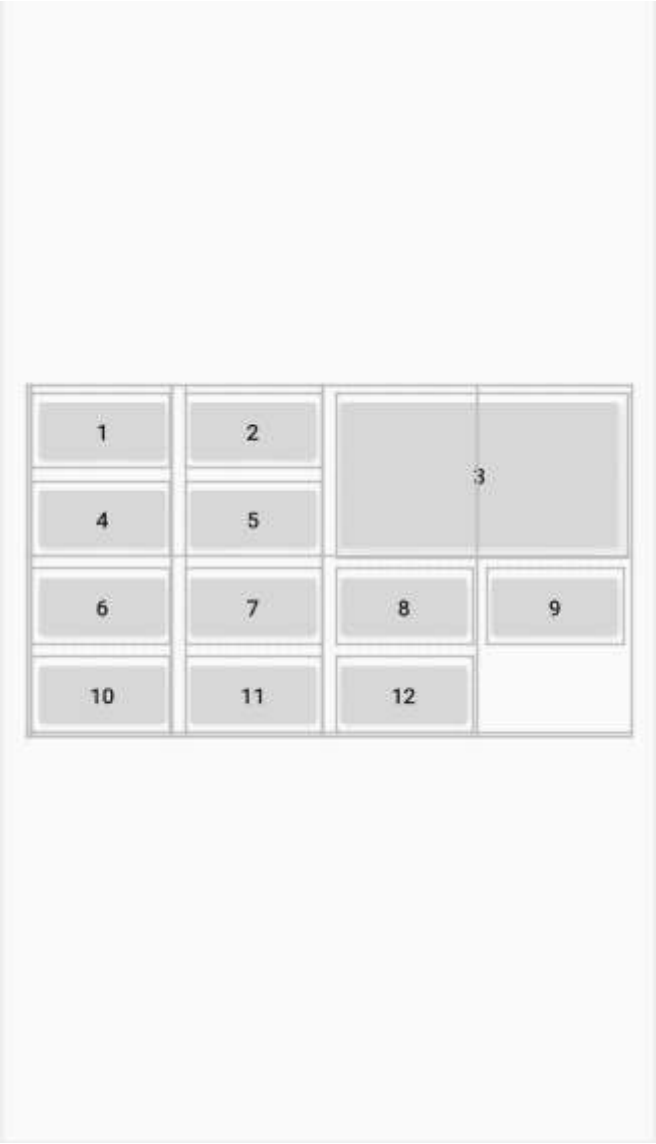
```
android:layout_gravity="fill_vertical"

android:layout_gravity="fill_horizontal"
```

komutlarını kullanabiliriz.

Her iki yönü de dolduracaksak,

```
android:layout_gravity="fill"
```

kodu yeterli olmaktadır.



rowSpan ve columnSpan kullanımı

Temel GridLayout kullanımını bu şekilde yapıyoruz.

Dinamik GridLayout ve gelişmiş özellikler için Alper Beyler'in anlatımını tavsiye ediyorum:

Android GridLayout Kullanımı | Android Dersleri | Mobilhanem

Merhaba Arkadaşlar, Mobilhanem.com sitemiz üzerinden anlattığımız/yayınladığımız Android Eğitimleri yazı serimizde bu...

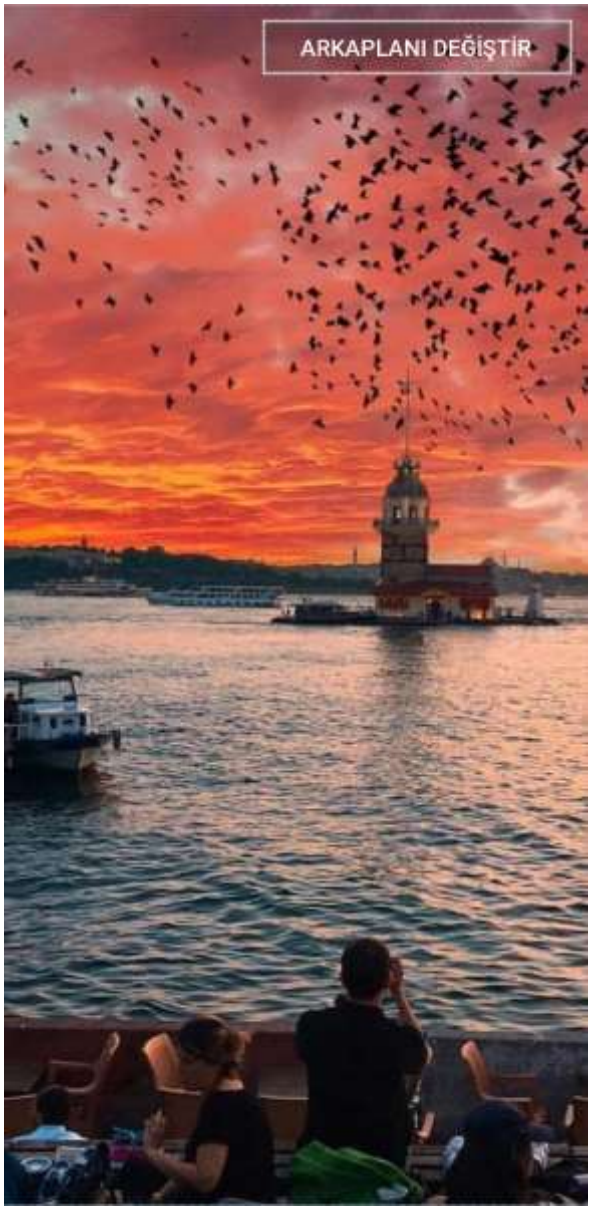
www.mobilhanem.com

4) FrameLayout

Frame Layout, yaygın olarak componentlerin birbirlerinin yerine geçebilmesine ve üst üste kullanabilmesine olanak sağlar. Bu şekilde yapıları çakışmadan birbirinin yerine kullanmamıza ve hangisinin üstte olacağına karar vermemizi sağlar.

FrameLayout componentlerin yerleşiminde stack gibi davranır.

Stack ise Last In First Out mantığıyla çalışır. Burada da FrameLayout'a eklenen en son bileşen en önde gözükecektir.



FrameLayout kullanımı — 1

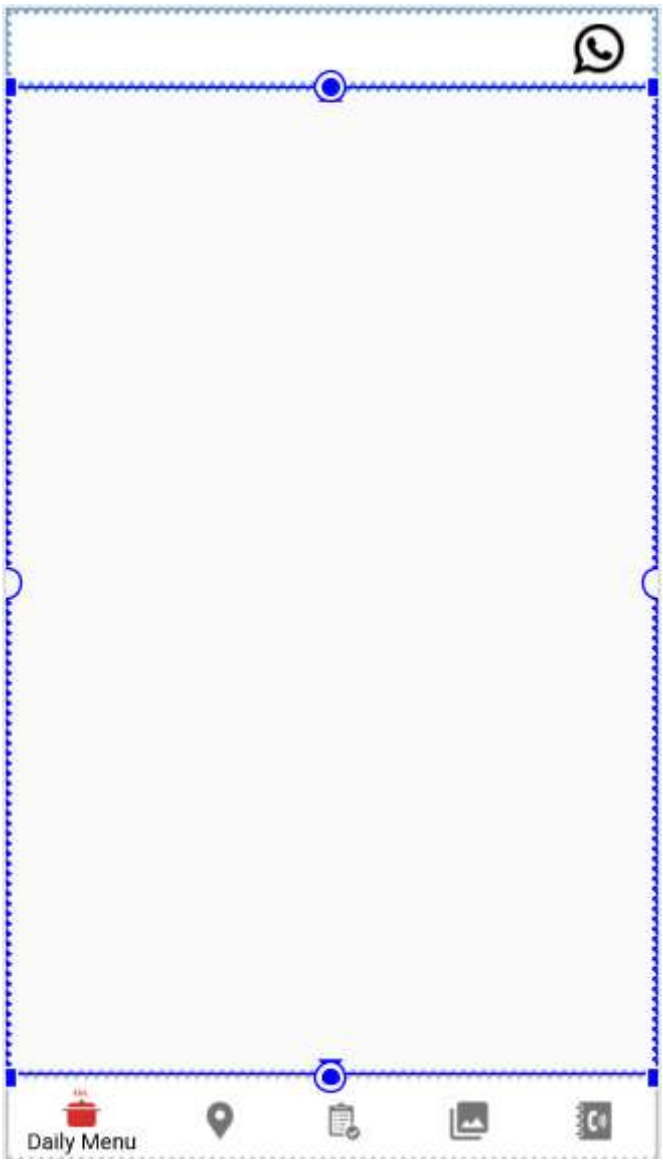
```
<?xml version="1.0" encoding="utf-8"?>
 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent" android:layout_height="match_parent">

     <ImageView
         android:id="@+id/setBackground"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         tools:srcCompat="@drawable/istanbul" />

```

```
12        <Button
13            android:id="@+id/changeImage"
14            android:layout_width="wrap_content"
15            android:layout_height="wrap_content"
16            android:layout_gravity="center_horizontal|right"
17            android:background="@null"
18            android:text="Arkaplanı Değiştir"
19            android:textColor="#FFFFFF" />
20
21    </FrameLayout>
```

frameLayout.xml hosted with ♡ by GitHub view raw

Ayrıca FrameLayoutlar sık olarak Fragmentlar ile kullanılsa da aralarında direk bir bağlantı yoktur. Burada da aynı şekilde fragmentların üstlerine binmeden birbiri arasında geçiş yapabilmesi söz konusudur.

Fragment başka bir konuya ait olduğu için bu kısma giriş yapmayacağım. *Sadece basit bir örnek göstermek istiyorum.*



FrameLayout Kullanımı — 2

Gördüğünüz bu tasarımda, alt kısımda Bottom Navigation Bar bulunmaktadır ve 5 tane farklı navigation vardır.

Kenarlarını mavi olarak işaretlediğimiz alan FrameLayout'tur ve Bottom Bar'da bulunan 5 navigation fragment yapısı ile FrameLayout üzerinde değişecektir. Buda günlük hayatta kullandığımız mobil uygulamalarda yaygın bir kullanımdır.

Dinamik FrameLayout ve gelişmiş özellikler için Alper Beyler'in anlatımını tavsiye ediyorum:

Android FrameLayout Kullanımı | Android Dersleri | Mobilhanem

Merhaba Arkadaşlar, Mobilhanem.com sitemiz üzerinden anlattığımız/yayınladığımız Android Eğitimleri yazı serimizde bu...

www.mobilhanem.com

5) ConstraintLayout

ConstraintLayout, Layout ailesinin yeni üyelerindendir ve Android Studio 2.2 sürümü ile gelmiştir.

Temel olarak, kullanıcıları karışık tasarımlardan kurtarmayı ve layout kullanımını minimuma indirmeyi hedefler. Sağladığı özellikler sayesinde componentleri birbirine göre hizalar ve herhangi birinin konumu, boyutu değiştiğinde diğer componentlerin de ona göre konum alabilmesine olanak tanır. Yani siz oturttuğunuz tasarımda ufak bir değişiklik yapmak istediğinizde tasarımın tamamını değiştirmek zorunda kalmazsınız. Diğer componentler haliyle konumunu düzeltecektir. Android'in ConstraintLayout ile amaçladığı budur.

Bileşenleri birbirine göre hizalamak, elle yapılabilmektedir. Fakat Android Studio'nun bunu yapmasını istediğinizde "Infer Constraints" seçeneğine tıkladığında da program tarafından gerçekleştirilebilir.

Çıkışında oldukça ses getirdiğinden hakkında çok sayıda video ve makale yapıldı. Beğendiğim 2 yazıyı paylaşmak istiyorum:

Android Studio 2.2' de ConstraintLayout Yapısı | Tuğba

Arayüze sahip yazılımsal projelerde (Web Yazılım,Ios ve Android Yazılımlar vb.) , projenin en az backend yazılım bölümü...

tugbaustundag.com

Derinlemesine Constraint Layout

Constraint Layout I/O 2016 da birçok güzelliği ve kolaylığı ile hayatımıza girdi, Google in dökümanları çok güzel olsa...

medium.com

Yazıma son vermeden önce, projelerinizi parça parça yönetmenizi kolaylaştıracak olan parçalı layout mantığını basit bir örnekle anlatmak istiyorum.

Örneğin, biz bir proje yapıyoruz ve anasayfasını tasarlayacağız. Anasayfa, projeler için çok önemlidir. Kullanıcıyı karşılar ve kullanıcı ulaşabileceği seçeneklere en kolay ve en az “tık” ile ulaşmak ister. Bununla beraber, tüm seçenekleri ulaştırmak adına kullanıcıyı anasayfa butonlara boğmak pek mantıklı değildir.

Anasayfamızın üstünde toolbar olsun.

Toolbar hakkında bilgi almak isterseniz;

Android Toolbar Kullanımı

Merhaba arkadaşlar, Bugün sizlere Lolipop (Android 5.0) güncellemesi ile gelen Android Toolbar yapısını nasıl...

www.mobilhanem.com

Toolbar’ı direk anasayfamıza monte edebiliriz fakat bunun yerine ayrı bir sayfada tasarlayıp anasayfamıza dahil etmek daha mantıklı olacaktır. İleride yapacağımız değişikliklerde de bize tasarımsal açıdan kolaylık sağlayacağı kesin.

Bir toolbar tasarlayalım.

toolbar.xml

Toolbar

Bunu Anasayfamıza dahil edelim.

```xml
<?xml version="1.0" encoding="utf-8"?>
 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:app="http://schemas.android.com/apk/res-auto"
     xmlns:tools="http://schemas.android.com/tools"
     android:id="@+id/container"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     tools:context=".ui.activity.MainActivity">

     <include
         android:id="@+id/toolbar"
         layout="@layout/toolbar"
         android:background="@color/toolbarColor"
         app:layout_constraintEnd_toEndOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent"></include>

 </RelativeLayout>
```

relativeLayout_toolbar.xml hosted with ♡ by GitHub view raw

ve taslağımızın son durumu:

Parçalı Tasarım

Taslağımızda hem Toolbar, hem FrameLayout hem de Bottom Navigation Bar kullanmış olduk.

Parça parça anlattığım bu yapıları 2. yazı ile tek proje üstünden anlatmayı hedefliyorum.

Faydalı olması dileğiyle.

Relativelayout Linearlayout Constraintlayout Android App Development Android Design