# آموزش برنامه نویسی اندروید
# در محیط اندروید استودیو

## آموزش کار با GridLayout

مدرس : سیدمهدی مطهری

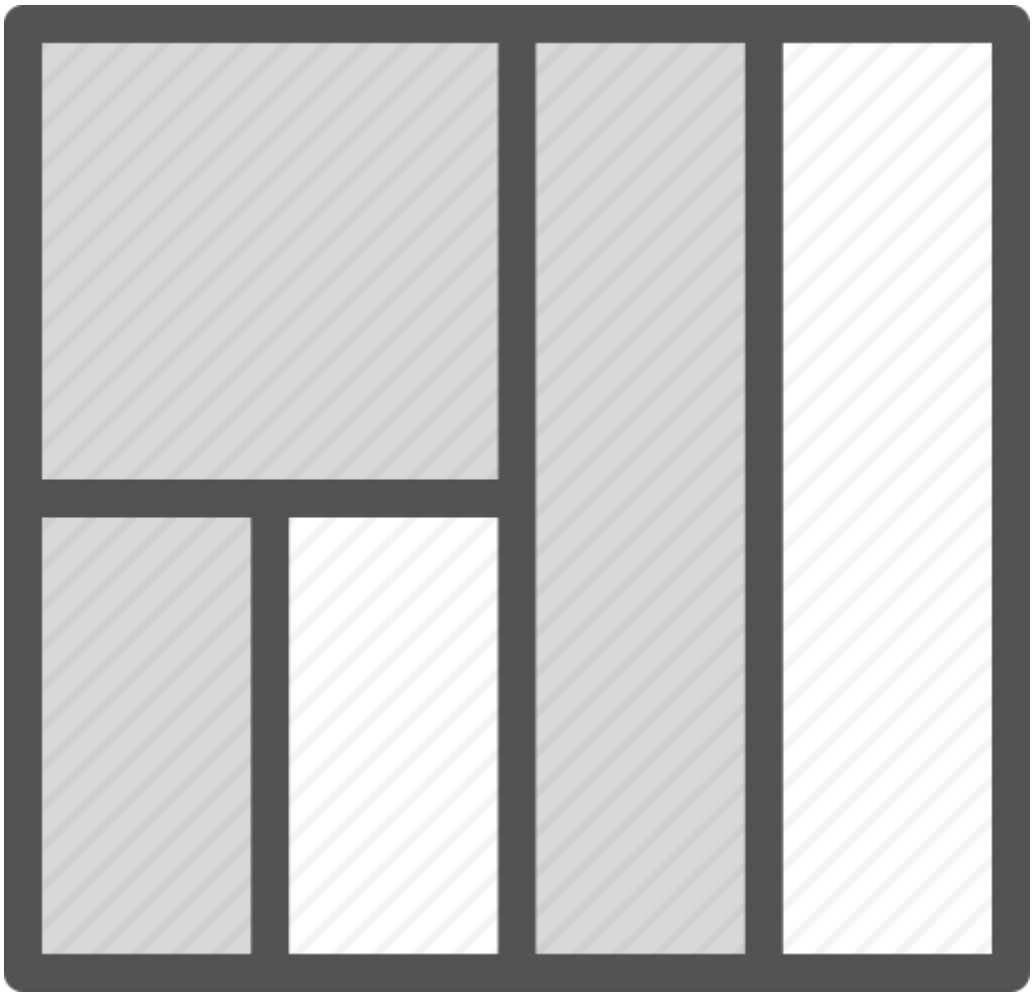
www.android-studio.ir

# به نام خدا

در جلسات گذشته با ViewGroup هایی مانند RelativeLayout، LinearLayout و TableLayout آشنا شدیم و توانستیم رابط‌های کاربری متفاوتی را توسط این Layout ها ایجاد کنیم. در این مبحث قصد دارم یک ViewGroup دیگر با نام GridLayout را معرفی کنم.

## GridLayout چیست؟

در اندروید ۴.۰ (API 14) یک ViewGroup جدید با نام GridLayout معرفی شد. Grid به معنای طراحی شبکه‌ای و سلولی است. مانند یک جدول که از تعدادی سطر و ستون تشکیل شده.



البته اصطلاح GridLayout منحصر به سیستم عامل اندروید نیست و اگر این واژه را در گوگل جستجو کنید به مثال‌هایی در سایز زمینه‌ها مانند CSS در طراحی صفحات وب برخورد خواهید کرد.

اگر مبحث TableLayout را فراموش کرده‌اید مجدد یک مرور کلی روی آن داشته باشید تا تفاوت بین این دو را بهتر درک کنید. در TableLayout صرفا امکان مدیریت View ها (به عبارتی، سلول‌ها)ی موجود در یک سطر را داشتیم و می‌توانستیم چند سطر یا ردیف زیر یکدیگر ایجاد کنیم. اما در GridLayout مدیریت ستون‌ها نیز امکان‌پذیر است. به عنوان مثال می‌توان ستون‌های دو سطر را با یکدیگر ادغام کرد. یک View مانند Button می‌تواند یک یا چند سلول را در جهت افقی یا عمودی اشغال کند. همچنین می‌توان یک سلول را از یک سطر به سطر دیگر (و از یک ستون به ستون دیگر) منتقل کرد.

علی‌رقم کاربردهای فراوان این ViewGroup، در منابع آموزشی کمتر به آن پرداخته شده که در این مبحث سعی می‌کنم تا حد امکان توضیحات کامل ارائه دهم.

## مزایای GridLayout

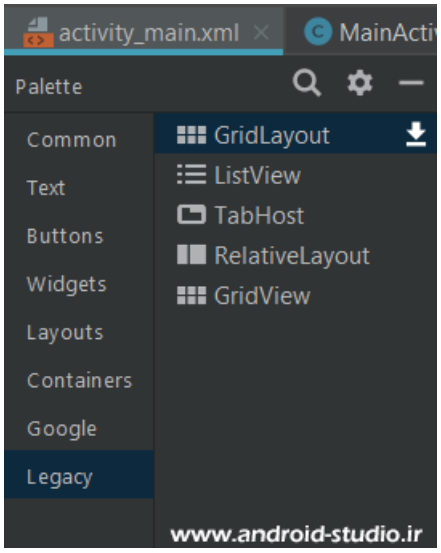
GridLayout عمدتا در طراحی لایه‌های پیچیده کاربرد دارد. لایه‌هایی که پیاده سازی آنها توسط سایر ViewGroup ها سخت و در مواردی غیر ممکن است. Layout ای که با GridLayout به راحتی ساخته می‌شود در سایر ViewGroup ما را با یک Layout تو در تو و عمیق مواجه می‌کند که این عمیق بودن خود باعث کاهش کارایی (Performance) برنامه می‌شود. همانطور که قبلا اشاره شد، در اینجا مدیریت ستون‌ها نیز به سادگی امکان‌پذیر است و به طور کلی انعطاف بیشتری در پیاده‌سازی لایه‌های پیچیده داریم.

اگر بخواهم ویژگی‌های این Layout را به صورت خلاصه بیان کنم، در سه گزینه خلاصه می‌شود:

- امکان کنترل و مدیریت سلول‌ها در هردو جهت سطر (Row) و ستون (Column)
- اجرای لایه‌های پیچیده بدون استفاده از Layout های تو در تو و در نتیجه بهینه بودن لایه
- ساخت ساده‌ی لایه‌هایی که از نظم و ترتیب مشخصی پیروی نمی‌کنند

## ساخت پروژه‌ی GridLayout

یک پروژه با نام GridLayout در اندروید استودیو با یک Empty Activity ایجاد می‌کنم و با ارائه‌ی چند مثال، به تشریح این ViewGroup می‌پردازم.




در پالت اندروید استودیو گزینه‌ی GridLayout قرار دارد که با کشیدن و رها کردن روی صفحه‌ی پیش نمایش، به لایه‌ی اکتیویتی اضافه می‌شود. اما من این کار را انجام نمی‌دهم زیرا این گزینه مربوط به کتابخانه Support است که با افزودن آن به پروژه، خط زیر به build.gradle اضافه می‌شود:

```
implementation 'com.android.support:gridlayout-v7:xx.x.x'
```

در رابط کاربری اکتیویتی نیز تگ XML زیر ایجاد می‌شود:

```
<android.support.v7.widget.GridLayout>

</android.support.v7.widget.GridLayout>
```

همانطور که در ابتدای مبحث اشاره شد، GridLayout در API 14 معرفی شد. استفاده از این کتابخانه در صورتی لازم است که اپلیکیشن ما روی دیوایس‌های API 14 به قبل نیز استفاده شود در صورتی که در زمان تهیه این آموزش، حدود ۱۰۰% دیوایس‌های اندرویدی بالای این نسخه هستند.

Layout موجود در activity_main.xml را حذف کرده و یک GridLayout جایگزین آن می‌کنم:

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

    <TextView
        android:text="Cell 1"/>

    <TextView
        android:text="Cell 2"/>

    <TextView
        android:text="Cell 3"/>

    <TextView
        android:text="Cell 4"/>

    <TextView
        android:text="Cell 5"/>

    <TextView
        android:text="Cell 6"/>

</GridLayout>
```

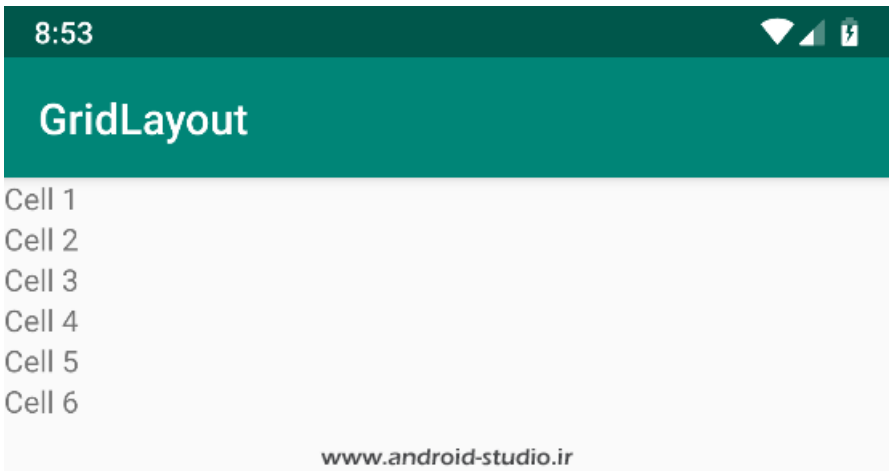
کد فوق، ساده ترین حالت است. در اینجا ۶ عدد TextView داریم که هرکدام نقش یک سلول را برای GridLayout ایفا می‌کند.

پروژه را روی شبیه ساز اجرا می‌کنم:


سلول‌ها به صورت افقی پشت یکدیگر قرار گرفته‌اند. در این ViewGroup نحوه چینش سلول‌ها را می‌توان در دو حالت horizontal (افقی) و vertical (عمودی) تعیین کرد. در قسمت قبل من خاصیت orientation را تعریف نکردم که به صورت پیش فرض مقدار horizontal برای آن درنظر گرفته شده.

مقدار vertical را اضافه می‌کنم:

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:text="Cell 1"/>

    <TextView
        android:text="Cell 2"/>

    <TextView
        android:text="Cell 3"/>

    <TextView
        android:text="Cell 4"/>

    <TextView
        android:text="Cell 5"/>

    <TextView
        android:text="Cell 6"/>

</GridLayout>
```




مشاهده می‌کنید سلول‌ها به صورت عمودی زیر یکدیگر قرار گرفتند.

در GridLayout دو خاصیت rowCount و columnCount به ترتیب برای تعیین حداکثر تعداد سطر و ستون استفاده می‌شود.

من مجدد حالت چینش را horizontal و برای columnCount مقدار 3 تعیین می‌کنم تا سلول‌ها در سه ستون قرار گیرند:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="3">

    <TextView
        android:text="Cell 1"/>

    <TextView
        android:text="Cell 2"/>

    <TextView
        android:text="Cell 3"/>

    <TextView
        android:text="Cell 4"/>

    <TextView
        android:text="Cell 5"/>

    <TextView
        android:text="Cell 6"/>

</GridLayout>
```


قبل از ادامه کار ابتدا یک استایل به TextView ها اضافه می‌کنم تا نتیجه تغییرات واضح‌تر باشد.

استایل زیر را در styles.xml ساختم:

```
<style name="Cells">
    <item name="android:background">#FF9800</item>
    <item name="android:padding">3dp</item>
    <item name="android:layout_marginLeft">3dp</item>
    <item name="android:layout_marginTop">3dp</item>
</style>
```

حالا استایل را به ویجت‌ها اضافه می‌کنم:

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="3">

    <TextView
        android:text="Cell 1"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 2"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 3"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 4"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 5"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 6"
        style="@style/Cells"/>

</GridLayout>
```


در حال حاضر چینش سلول‌ها به صورت افقی است. یعنی 3 سلول اول در سطر اول و 3 سلول دوم در سطر دوم. حالا می‌خواهم ترتیب قرارگیری به صورت ستونی باشد. Orientation را تغییر داده و rowCount را با مقدار 2 اضافه می‌کنم تا تعداد سطرها تغییر نکند:

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:columnCount="3"
    android:rowCount="2">

    <TextView
        android:text="Cell 1"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 2"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 3"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 4"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 5"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 6"
        style="@style/Cells"/>

</GridLayout>
```

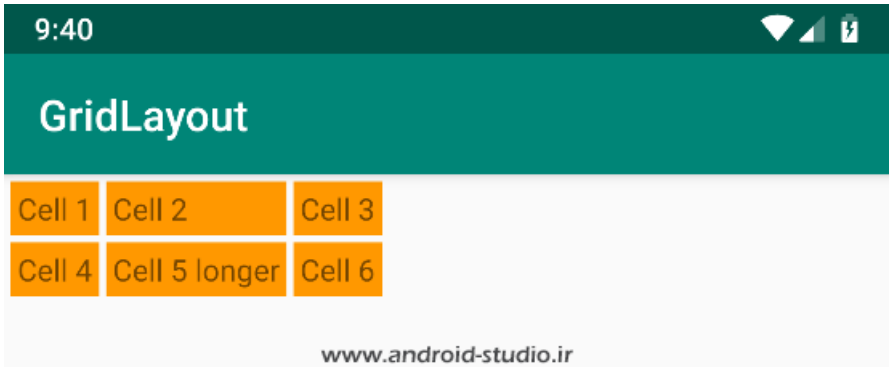

برای ادامه‌ی آموزش orientation را به حالت افقی برمی‌گردانم.

کلمه longer را به text پنجم اضافه می‌کنم تا طولانی‌تر شود:

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="3"
    android:rowCount="2">

    <TextView
        android:text="Cell 1"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 2"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 3"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 4"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 5 longer"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 6"
        style="@style/Cells"/>

</GridLayout>
```

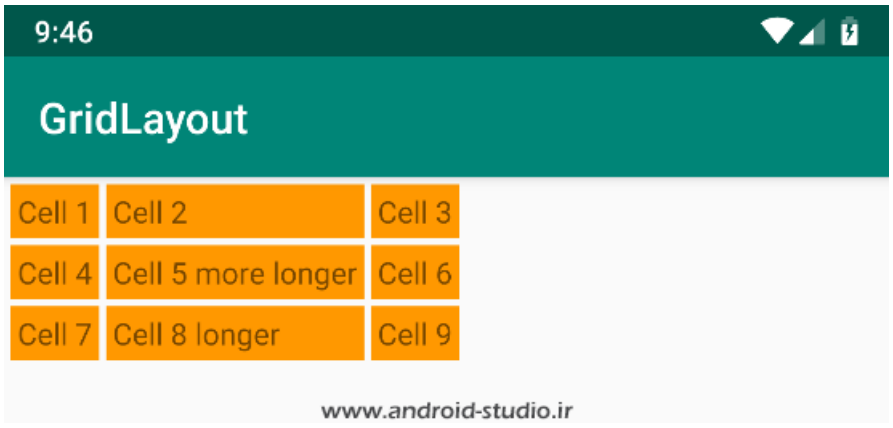

در ابتدای مبحث گفتیم یکی از مزایای GridLayout امکان مدیریت ستون‌هاست. در تصویر بالا ملاحظه می‌کنید طول Cell2 از سلول 5 کوتاهتر است و مقداری فضای خالی ایجاد شده که لازم است این فضای خالی پر شود. برای اینکار خاصیت layout_gravity با مقدار fill_horizontal را به سلول‌های این ستون اضافه می‌کنم:

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="3"
    android:rowCount="2">

    <TextView
        android:text="Cell 1"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 2"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"/>

    <TextView
        android:text="Cell 3"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 4"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 5 longer"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"/>

    <TextView
        android:text="Cell 6"
        style="@style/Cells"/>

</GridLayout>
```




یک ردیف دیگر اضافه می‌کنم تا نتیجه را بهتر درک کنید:

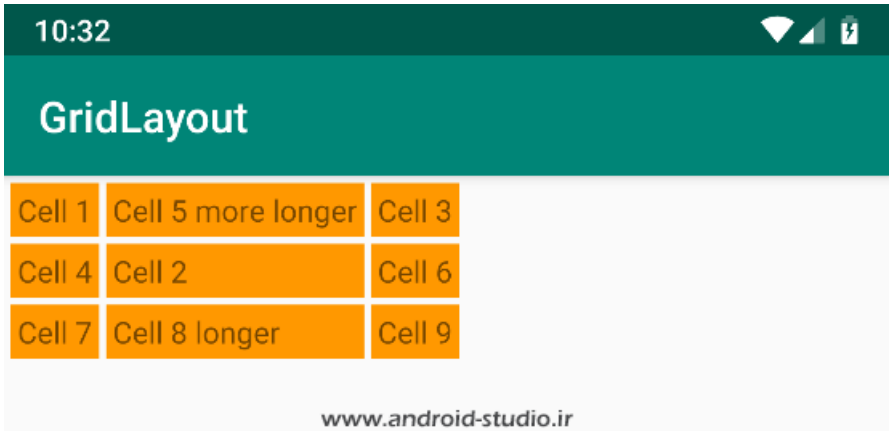
```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="3"
    android:rowCount="3">

    <TextView
        android:text="Cell 1"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 2"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"/>

    <TextView
        android:text="Cell 3"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 4"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 5 more longer"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"/>

    <TextView
        android:text="Cell 6"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 7"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 8 longer"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"/>

    <TextView
        android:text="Cell 9"
        style="@style/Cells"/>

</GridLayout>
```



تمام سلول‌های موجود در ستون دوم به اندازه‌ی بزرگترین سلول کشیده شده‌اند.

یکی دیگر از قابلیت‌های لایه‌ی گرید، امکان جابجایی و تغییر موقعیت سلول‌هاست. به عنوان مثال می‌توان مکان Cell 2 و Cell 5 را تغییر داد بطوری که هرکدام در جای دیگری قرار گیرد. این کار توسط دو خاصیت layout_row و layout_column انجام می‌شود. به کد و خروجی زیر دقت کنید:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="3"
    android:rowCount="3">

    <TextView
        android:text="Cell 1"
        style="@style/Cells" />

    <TextView
        android:text="Cell 2"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"
        android:layout_row="1"/>

    <TextView
        android:text="Cell 3"
        style="@style/Cells"
        android:layout_row="0"/>

    <TextView
        android:text="Cell 4"
        style="@style/Cells" />

    <TextView
        android:text="Cell 5 more longer"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"
        android:layout_row="0"
        android:layout_column="1"/>

    <TextView
        android:text="Cell 6"
        style="@style/Cells"
        android:layout_row="1"/>

    <TextView
        android:text="Cell 7"
        style="@style/Cells"/>

    <TextView
        android:text="Cell 8 longer"
        style="@style/Cells"
        android:layout_gravity="fill_horizontal"/>

    <TextView
        android:text="Cell 9"
        style="@style/Cells"/>

</GridLayout>
```



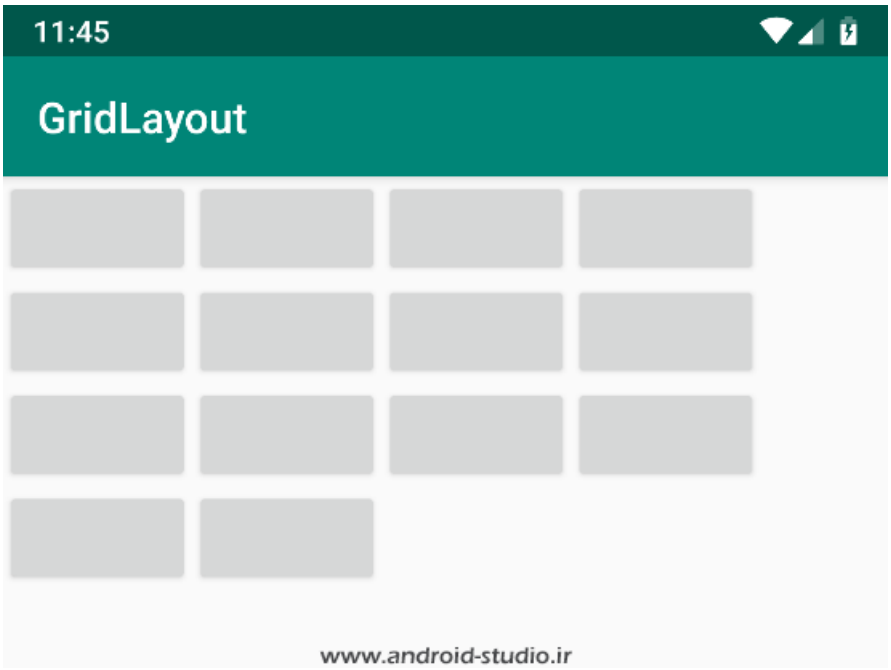

**نکته:** شمارش سطر و ستون از عدد 0 آغاز می‌شود. یعنی موقعیت سلول در سطر اول/ستون اول برابر است با (0.0)

ابتدا برای Cell 2 خاصیت layout_row با مقدار ۱ تعریف کردم تا این سلول در سطر دوم قرار گیرد. با انجام این تغییر، Cell 3 نیز به سطر دوم منتقل می‌شود که با تعریف layout_row برای این سلول و قرار دادن مقدار ۰، مجدد به سطر اول باز می‌گردد. در ادامه موقعیت Cell 5 را تعیین می‌کنم. مقدار ۰ برای layout_row و مقدار ۱ برای layout_column این سلول را در سطر اول/ستون دوم قرار می‌دهد. مانند مرحله قبل که با جابجایی Cell 2 سلول بعد از آن یعنی Cell 3 نیز جابجا شد، در اینجا هم Cell 6 به همراه Cell 5 جابجا می‌شود که با layout_row و مقدار ۱ مجدد آنرا به سطر دوم باز می‌گردانم. اینجاست که انعطاف پذیری بالای این ViewGroup ثابت می‌شود.

یک Activity دیگر از نوع Empty Activity با نام Calculator به پروژه اضافه می‌کنم. هنگام ساخت اکتیویتی، Launcher Activity را انتخاب می‌کنم تا به عنوان اکتیویتی اصلی در شبیه ساز اجرا شود. همانطور که از نام اکتیویتی پیداست، در این مرحله با یک ماشین حساب سروکار داریم.

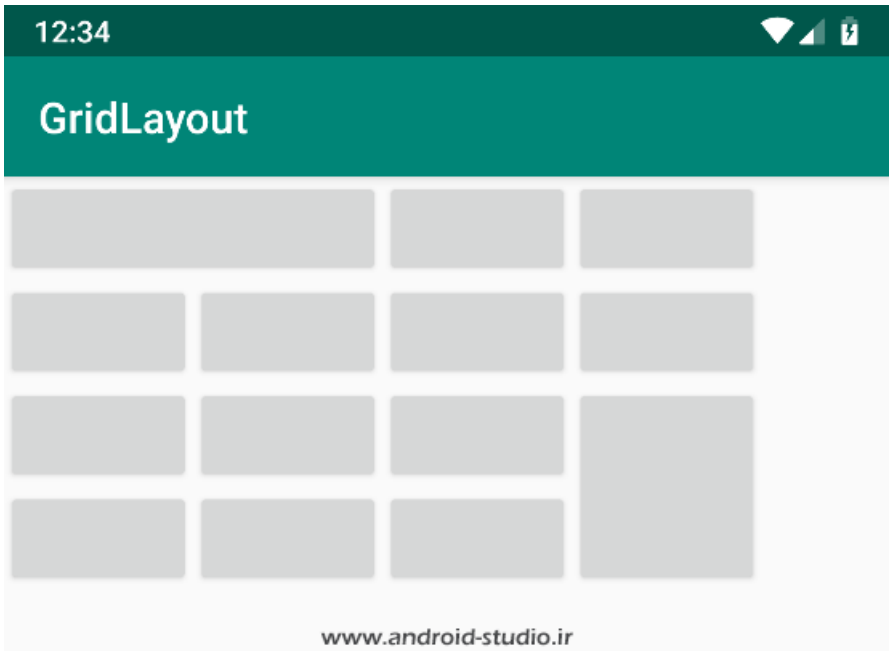
قصد دارم دکمه‌های این ماشین حساب را با GridLayout پیاده سازی کنم. ۱۴ دکمه که در یک Grid با ۴ سطر و ۴ ستون قرار گرفته‌اند.

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="4"
    android:rowCount="4">

    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />

</GridLayout>
```




کاری که باید انجام دهم این است که دو دکمه را به نحوی ویرایش کنم که هرکدام به اندازه‌ی دو دکمه ی معمولی جا بگیرد. یکی در جهت افقی و دیگری در جهت عمودی. در اینجا از دو خاصیت layout_rowSpan و layout_columnSpan استفاده می‌کنم. لغت Span به معنی طول است بنابراین خاصیت اولی برای تعیین طول سطر و دومی برای طول ستون بکار می‌رود.

```xml
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:columnCount="4"
    android:rowCount="4">

    <Button
        android:layout_columnSpan="2"
        android:layout_gravity="fill"/>
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button />
    <Button
        android:layout_rowSpan="2"
        android:layout_gravity="fill"/>
    <Button />
    <Button />
    <Button />

</GridLayout>
```




برای دکمه اول خاصیت layout_columnSpan با مقدار ۲ تعیین کردم. یعنی این ویجت باید به اندازه‌ی دو ستون فضا در اختیار داشته باشد. سپس توسط layout_gravity و مقدار fill، دکمه تمام فضای خالی موجود را اشغال می‌کند. به همین ترتیب برای دکمه‌ی یازدهم layout_rowSpan و layout_gravity تعریف می‌کنم که در نهایت این دکمه به اندازه دو سطر طول دارد.

**نکته:** در این مثال من بجای fill_horizontal و fill_vertical تنها از fill استفاده کردم که در هردو مشترک است و تفاوتی نمی‌کند از کدام مقدار استفاده کنیم.

**تمرین:** سعی کنید مثال فوق را توسط LinearLayout یا RelativeLayout و یا TableLayout پیاده سازی کرده، سپس حجم کد XML و زمانی که برای هردو صرف شده را مقایسه کنید.

در مثال آخر یک فرم ورود (Login Form) را توسط GridLayout طراحی می‌کنم.

یک اکتیویتی دیگر با نام Form به پروژه اضافه می‌کنم.

ابتدا کد و خروجی شبیه ساز را درج کرده سپس به توضیح جزئیات می‌پردازم:

:activity_form.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4">

    <TextView
        android:layout_row="0"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_gravity="center_horizontal"
        android:text="ورود"
        android:textSize="30sp" />

    <TextView
        android:layout_row="1"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_gravity="right"
        android:text="لطفا نام کاربری و رمز عبور را وارد کنید :"
        android:textSize="17sp" />

    <Space
        android:layout_row="2"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_height="30dp" />

    <TextView
        android:layout_row="3"
        android:layout_column="0"
        android:layout_gravity="right"
        android:text="نام کاربری" />

    <EditText
        android:ems="10" />

    <TextView
        android:layout_row="4"
```

```
        android:layout_column="0"
        android:layout_gravity="right"
        android:text="رمز عبور" />

    <EditText
        android:ems="10" />

    <Space
        android:layout_row="5"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_height="30dp" />

    <Button
        android:layout_row="6"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_gravity="center_horizontal"
        android:text="ورود" />

    <Space
        android:layout_row="7"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_gravity="fill" />

    <Button
        android:layout_row="8"
        android:layout_column="0"
        android:layout_columnSpan="4"
        android:layout_gravity="fill_horizontal"
        android:text="رمز عبور را فراموش کرده ام" />

</GridLayout>
```


در GridLayout برای ایجاد فاصله بین سلول‌ها بجای خاصیت margin از تگ Space استفاده می‌کنیم. در واقع Space خود به عنوان یک سلول به Grid اضافه می‌شود.

چند خاصیت دیگر برای این ViewGroup تعریف شده که useDefaultMargins یکی از آنهاست. این خاصیت یک فاصله‌ی پیش فرض بین سلول‌ها ایجاد می‌کند. البته در صورتی که قبلا فاصله‌ای تعریف نکرده باشیم.

خط زیر را به تگ GridLayout اضافه می‌کنم:

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


اسکرین شات فوق را با تصویر قبل مقایسه کنید. اندکی فضای خالی مابین سلول‌ها ایجاد شده.

سایر ویژگی‌ها را صرفا معرفی می‌کنم:

| خاصیت | توضیح |
|---|---|
| android:columnOrderPreserved | اگر مقدار "true" تعیین شود عرض ستون بر اساس محتوای درون آن تنظیم می‌گردد ولی چنانچه "false" تعیین شود به نحوی تنظیم می‌شود تا سایر ستون‌ها نیز در وضعیت مناسبی قرار گیرند. |
| android:rowOrderPreserved | اگر مقدار "true" تعیین شود ارتفاع سطر بر اساس محتوای درون آن تنظیم می‌گردد ولی چنانچه "false" تعیین شود به نحوی تنظیم می‌شود تا سایر سطرها نیز در وضعیت مناسبی قرار گیرند. |

**مطالعه‌ی بیشتر:**

https://developer.android.com/reference/android/widget/GridLayout

https://developer.android.com/reference/android/widget/GridLayout.LayoutParams.html

**توجه:** سورس پروژه درون پوشه Exercises قرار دارد

با ارائه انتقادات و پیشنهادات خود، ما را در ارائه آموزش‌های بهتر یاری فرمائید.
**www.android-studio.ir**