توسعه برنامه های موبایل

جلسه پنجم مجازی

بخش دوم

سحر صادقی

Life cycle

رویداد LifeCycle **وابسته باز بودن در پنجره مربوطه می باشد**

یک View **در یک سلسله مراتب به پنجره جاری پیوسته است** View . **بسته به یک** Lifecycle **است**.

متد onAttachedToWindow() **زمانی فراخوانی می شود که پنجره در حافظه بارگذاری شده و در دسترس باشد**.

متد onDetachedFromWindow() **زمانی فراخوانی می شود که** view **از میزبان** (parent) **خود جدا شده باشد (و البته میزبان نیز خود به یک** window **وصل باشد). این اتفاق، برای مثال، زمانی رخ می دهد که** activity **به عنوان مثال با صدا خورده شدن متد** finished() **یا** view **بازیافت شده باشند**.

متد onAttachedToWindow() **زمانی فراخوانی می شود که یک پنجره در دسترس باشد**.

متد onDetachedFromWindow() **از والد** View **پس از حذف استفاده می نماید(حتی اگر به یک پنجره دیگر پیوست باشد) . این رویداد نیز برای** Activity **به صورت باز پس گیری (حتی وقتی متد** finished() **فراخوانی شده باشد(روی می دهد**.

متد onDetachedFromWindow() **را می توان جهت متوقف کردن انیمیشن ها و پاک سازی و آزاد نمودن منابع مورد استفاده ی** view **فراخوانی نمود**.

Event**های مربوط به** life cycle **به صورت ترتیبی/پیمایشی**(Traversal life cycle event)

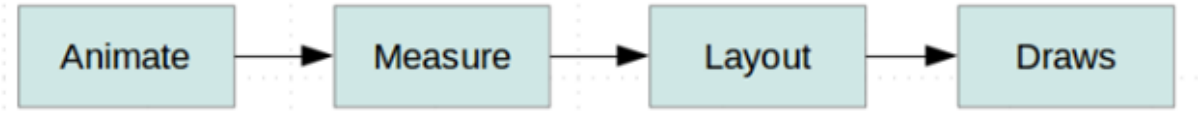
رخدادهای چرخه ی حیات به ترتیب عبارت اند از:

Animate

Measure

Layout

Draw



لازم است view **ها با نحوه ی اندازه گیری و چیدمان** (layout) **خود آشنا باشند. متد** requestLayout() **به** view**اعلان می کند که خود را اندازه گرفته و موقعیت دهی** (layout) **نمایند. از آنجایی که این عملیات ممکن است** layout **دیگر** view **ها را نیز متاثر کند، متد** requestLayout() **پدر** (parent) **نیز فراخوانی می شود.**

نکته:

زمانی که چندین layout **را داخل هم قرار داده و آن ها را زیاد تودرتو می نمایید، این امر سبب به اجرا در آمدن الگوریتم فراخوانی بازگشتی می شود. حال زمانی که عمق ساختار درختی یا سلسله مراتب زیاد بوده و در این حین لازم باشد سلسله مراتب مجددا محاسبه شود، عملیات اندازه گیری و موقعیت دهی ناگذیر سنگین و طولانی خواهد بود.**

متد onMeasure() **اندازه ی** view **و المان های محصور آن** (view **های فرزند) را مشخص می کند. سپس با فراخوانی** setMeasureDimension() **داخل بدنه ی خود و قبل از اجرای دستور** return **، اندازه ی آن ها را تنظیم می نماید.**

متد onLayout() **تمامی** view **ها را بر اساس نتیجه یا خروجی متد** onMeasure() **موقعیت دهی می نماید. این فراخوانی معمولا تنها یکبار انجام می شود در حالی که** onMeasure() **می تواند چندین بار صدا خورده شود.**

چرخه ی حیات activity

View**ها به** event **های چرخه ی حیات** activity **ها دسترسی ندارند. اگر لازم باشد که** view **ها از رخداد این** event **ها باخبر شوند، در آن صورت می بایست یک** interface **در** view **ایجاد نمایید** (interface **مربوطه را در بدنه ی کلاس** view **پیاده سازی کرده) و متدهای مربوط به مدیریت چرخه ی حیات** activity **را از طریق آن فراخونی نمایید.**

تعریف attribute های بیشتر برای view های اختصاصی

شما می توانید attribute **های اضافی بر سازمان برای** view **های ترکیبی و اختصاصی خود تعریف نمایید. برای این منظور ابتدا بایستی یک فایل به نام** attrs.xml **در پوشه ی** res/values **خود ایجاد نمایید. در زیر مثالی را می بینید که برای** view **جدیدی به نام** ColorOptionsView **تعدادی** attribute **جدید تعریف شده است.**

```
1<!--?xml version="1.0" encoding="utf-8" ?-->
2<resources>
```

```
<declare-styleable name="ColorOptionsView">
<attr name="titleText" format="string" localization="suggested">
<attr name="valueColor" format="color">
    </attr></attr></declare-styleable>
 </resources>
```

جهت استفاده از attribute **های نام برده در فایل**layout **، می بایست آن ها را در بخش** header **فایل** XML **اعلان نمایید. در کد زیر این کار توسط دستور** xmlns:custom **صورت گرفته است. مقدار** attribute **هایی که زیر این دستور درج می شوند، متعاقبا به** view **مورد نظر نیز اعمال می شوند.**

?

```
<linearlayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools" <!--="" define="" new="" name="" space=""
for="" your="" attributes="" --="">
    xmlns:custom="http://schemas.android.com/apk/res/com.vogella.android.view.compoundview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
>
<!-- Assume that this is your new component. It uses your new attributes -->
        <com.vogella.android.view.compoundview.coloroptionsview
android:layout_width="match_parent"
android:layout_height="?android:attr/listPreferredItemHeight" custom:titletext="Background
color" custom:valuecolor="@android:color/holo_green_light">
</com.vogella.android.view.compoundview.coloroptionsview></linearlayout>
```

مثال زیر نشان می دهد چگونه کامپونت های شما می توانند به این attribute **ها دسترسی داشته باشند.**

?

تمرین: ساخت و پیاده سازی یک view ترکیبی

```
1
2  package com.vogella.android.view.compoundview;
3  import android.content.Context;
4  import android.content.res.TypedArray;
5  import android.util.AttributeSet;
6  import android.view.Gravity;
7  import android.view.LayoutInflater;
8  import android.view.View;
9  import android.widget.ImageView;
10 import android.widget.LinearLayout;
11 import android.widget.TextView;
12 public class ColorOptionsView extends View {
13         private View mValue;
14         private ImageView mImage;
15         public ColorOptionsView(Context context, AttributeSet attrs) {
16                 super(context, attrs);
17                 TypedArray a = context.obtainStyledAttributes(attrs,
18                                 R.styleable.Options, 0, 0);
19                 String titleText = a.getString(R.styleable.Options_titleText);
20                 int valueColor =
   a.getColor(R.styleable.Options_valueColor,                    android.R.color.holo_bl
21                 a.recycle();
22                 // more stuff
23
```

ایجاد پروژه

یک پروژه ی جدید اندروید با داده ها و مقادیر زیر ایجاد نمایید.

جدول پروژه ی جدید اندرویدی (۱,۱)

Property	Value

)Testing(**تست**

Table width

)Application Name(**اسم اپلیکیشن**

Compound view example

)Project Name(**اسم پروژه**

com.vogella.android.customview.compoundview

)Package name(**اسم پوشه ی حاوی کلاس ها**

com.vogella.android.customview.compoundview

API (Minimum, Target, Compile with)**پایین ترین ویرایش اندروید که برنامه بایستی بر روی آن قابل اجرا باشد، ویرایشی که برنامه برای آن طراحی شده، ورژنی که برنامه با آن کامپایل می شود**

Latest

)Template(**قالب آماده و پیاده سازی پروژه**

Empty Activity

Activity

MainActivity

Layout

activity_main

تعریف و استفاده از attribute های جدید

یک فایل حامل attribute **های جدید به نام** attr.xml **را داخل پوشه ی** res/values **ایجاد نمایید**.

?

```
1<!--?xml version="1.0" encoding="utf-8" ?-->
2<resources>
3<declare-styleable name="Options">
4<attr name="titleText" format="string" localization="suggested">
5<attr name="valueColor" format="color">
6    </attr></attr></declare-styleable>
7</resources>
```

محتوای فایل layout **ای که توسط کلاس** activity **برای نمایش در** UI **فراخوانی می شود را به صورت زیر ویرایش نمایید.**

?

```
<linearlayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
xmlns:custom="http://schemas.android.com/apk/res/com.vogella.android.view.com
poundview" android:layout_width="match_parent"
android:layout_height="match_parent" android:orientation="vertical"
android:showdividers="middle" android:divider="?android:attr/listDivider"
tools:context=".MainActivity">

<com.vogella.android.view.compoundview.coloroptionsview
android:id="@+id/view1" android:layout_width="match_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
android:background="?android:selectableItemBackground"
android:onclick="onClicked" custom:titletext="Background color"
custom:valuecolor="@android:color/holo_green_light">

<com.vogella.android.view.compoundview.coloroptionsview
android:id="@+id/view2" android:layout_width="match_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
android:background="?android:selectableItemBackground"
android:onclick="onClicked" custom:titletext="Foreground color"
custom:valuecolor="@android:color/holo_orange_dark">

</com.vogella.android.view.compoundview.coloroptionsview></com.vogella.androi
d.view.compoundview.coloroptionsview></linearlayout>
```

view_color_options**را برای** view **اختصاصی خود ایجاد نمایید.**

?

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

<merge xmlns:android="http://schemas.android.com/apk/res/android">
1
<textview android:layout_width="0dp" android:layout_height="wrap_content"
2android:layout_weight="1" android:layout_centervertical="true" android:layout_marginleft="16dp"
android:textsize="18sp">
3
<view android:layout_width="26dp" android:layout_height="26dp"
4android:layout_centervertical="true" android:layout_marginleft="16dp"
android:layout_marginright="16dp">
5
<imageview android:layout_width="wrap_content" android:layout_height="wrap_content"
6android:layout_marginright="16dp" android:layout_centervertical="true"
android:visibility="gone">

</imageview></view></textview></merge>
```

اختصاصی خود را به صورت زیر پیاده سازی نمایید. View

[?]

```
1package com.vogella.android.customview.compoundview;

2import com.vogella.android.view.compoundview.R;

3import android.content.Context;

4import android.content.res.TypedArray;

5import android.util.AttributeSet;

6import android.view.Gravity;

7import android.view.LayoutInflater;

8import android.view.View;

9import android.widget.ImageView;

10import android.widget.LinearLayout;

11public class ColorOptionsView extends LinearLayout {

12        private View mValue;

13        private ImageView mImage;

14        public ColorOptionsView(Context context, AttributeSet
attrs) {

15                super(context, attrs);

16                TypedArray a =
context.obtainStyledAttributes(attrs,

17                                R.styleable.ColorOptionsView,

180, 0);
```

```
19                  String titleText =
  a.getString(R.styleable.ColorOptionsView_titleText);
20
                    int valueColor =
21a.getColor(R.styleable.ColorOptionsView_valueColor,    android
  .R.color.holo_blue_light);
22
                    a.recycle();
23
                    setOrientation(LinearLayout.HORIZONTAL);
24
                    setGravity(Gravity.CENTER_VERTICAL);
25
                    LayoutInflater inflater = (LayoutInflater)
  context                                 .getSystemService(Context
26.LAYOUT_INFLATER_SERVICE);
27
                    inflater.inflate(R.layout.view_color_options,
  this, true);
28
                    TextView title = (TextView) getChildAt(0);
29
                    title.setText(titleText);
30
                    mValue = getChildAt(1);
31
                    mValue.setBackgroundColor(valueColor);
32
                    mImage = (ImageView) getChildAt(2);
33          }
34          public ColorOptionsView(Context context) {
                    this(context, null);
35          }
36          public void setValueColor(int color) {
37
                    mValue.setBackgroundColor(color);
38          }
39          public void setImageVisible(boolean visible) {
40
                    mImage.setVisibility(visible ? View.VISIBLE :
  View.GONE);
41          }
    }
```

تنظیم و ویرایشactivity

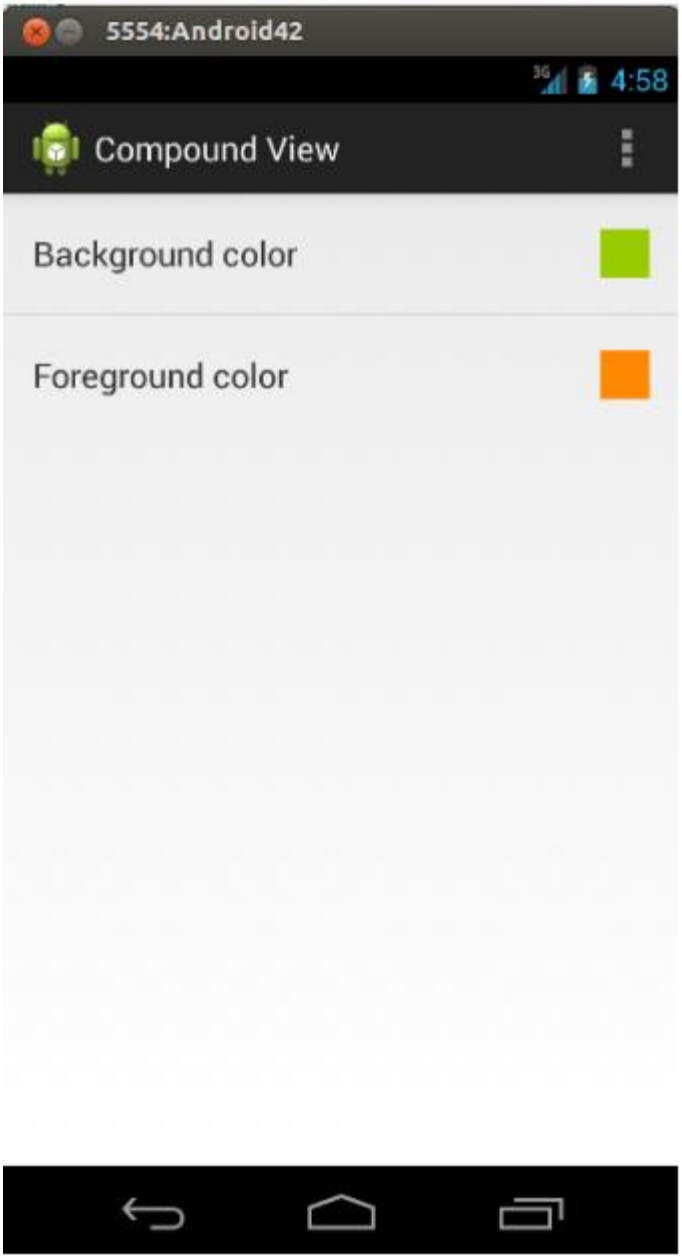
دنه ی activity **را به صورت زیر ویرایش نموده و سپس اپلیکیشن خود را اجرا نمایید.**

?

```
1package com.vogella.android.customview.compoundview;
2import com.vogella.android.view.compoundview.R;
```

```
3import android.app.Activity;
4import android.os.Bundle;
5import android.view.Menu;
6import android.view.View;
7import android.widget.Toast;
8public class MainActivity extends Activity {
9        @Override
10       protected void onCreate(Bundle savedInstanceState) {
11               super.onCreate(savedInstanceState);
12               setContentView(R.layout.activity_main);
13       }
14       @Override
15       public boolean onCreateOptionsMenu(Menu menu) {
16               // Inflate the menu; this adds items to the action
bar if it is present.
17               getMenuInflater().inflate(R.menu.activity_main,
menu);
18               return true;
19       }
20       public void onClicked(View view) {
21               String text = view.getId() == R.id.view1 ?
"Background" : "Foreground";
22               Toast.makeText(this, text,
Toast.LENGTH_SHORT).show();
23       }
24}
```

اپلیکیشن در زمان اجرا می بایست ظاهری مشابه زیر داشته باشد.



Canvas API) توابع مرتبط با کلاس(Canvas

شرح مفهوم Canvas API

Canvas API **به شما اجازه می دهد تا افکت های دیداری و گرافیکی منحصر بفردی خلق نمایید.**

کلاس Canvas **تعدادی تابع جهت ترسیم اشکال در اختیار دارد. برای انجام عملیات ترسیم، توسعه دهنده به ۴ مولفه ی ابتدایی احتیاج دارد: ۱. یک** bitmap **جهت میزبانی پیکسل ها ۲. یک** Canvas **برای نگه داشتن توابع**

Bitmap 4. ،text ،Path ، Rect**مانند ترسیم نوشتن و جهت کلاس یک** bitmap) 3. draw(**کلاس** paint **به منظور تعریف رنگ ها و** style **های المان های گرافیکی**.

در واقع شما بر روی سطح Bitmap **اشکال را ترسیم می کنید، توابع ترسیم اشکال را از کلاس** Canvas **وام گرفته و در نهایت با استفاده از کلاس** Paint **رنگ و استایل اشکال مورد نظر بر روی سطح** canvas **را تعیین می نمایید**.

کلاسCanvas

آبجکت Canvas **آن فایل تصویری پیکسلی یا** bitmap **ای که اشکال بر روی آن ترسیم می شوند را در برمی گیرد. علاوه بر آن توابعی را ویژه ی تنظیم رنگ**(drawARGB()) **، ترسیم عکس پیکسلی** (drawBitmap() **، درج متن**(drawText()) **، ترسیم مستطیل با گوشه های گرد** (drawRoundRect()) **و غیره ... در اختیار برنامه نویس قرار می دهد**.

کلاسPaint

برای تنظیم رنگ، فونت و غیره ... بر روی آبجکت Canvas **به یک آبجکت از جنس کلاس** Paint **نیاز دارید**.

کلاس Paint **در واقع این قابلیت را در اختیار شما قرار می دهد تا رنگ، فونت و برخی از افکت ها را برای انجام عملیات ترسیم مشخص نمایید**.

متد setStyle() **به شما اجازه می دهد تا به سیستم اعلان نمایید آیا تنها** outline **و خط پیرامون** (Paint.Style.STROKE**باید ترسیم شود یا تنها بخش رنگ شده** (Paint.Style.FILL) **و یا هر دو بخش مزبور**.

برای تنظیم کانال آلفا Paint **کافی است متد** setAlpha() **را فراخوانی نمایید**.

جهت تنظیم رنگ های بیشتر برای آبجکت Paint() **می توانید از** Shaders **استفاده نمایید**.

Shader

آبجکت shader **این امکان را فراهم می کند تا برای آبجکت** Paint **محتوایی که باید ترسیم شوند را مشخص نمایید. برای مثال، می توانید با استفاده از** BitmapShader **که کلاس مشتق از** Shader **هست، یک عکس پیکسلی** (Bitmap) **تعریف نمایید و سپس در آن شکل هندسی دلخواه را ترسیم نمایید. بدین وسیله شما می توانید به عنوان نمونه، تصویری با گوشه های گرد بکشید. کافی است یک** BitmapShader **برای آبجکت** Paint **تعریف نموده و با فراخوانی تابع**drawRoundRect() **، شکل مستطیل را با گوشه های گرد ترسیم نمایید**.

محیط اندروید (platform) **تعدادی کلاس مشتق از** Shader **جهت ترسیم و پیاده سازی شیب رنگ ارائه می دهد. این کلاس ها عبارتند از**LinearGradient ،: RadialGradient**و**.SweepGradient

به منظور استفاده از Shader **لازم است آن را از طریق متد** ()setShader **به آبجکت** Paint **انتساب دهید**.

در صورتی که ناحیه ی پر شده بزرگ تر از Shaders **هست، آنگاه می توانید از طریق** Shader tile model **مشخص کنید که باقی ناحیه ی مربوطه چگونه پر شود. به طور مثال، ثابت** Shader.TileMode.CLAMP **به سیستم دستور می دهد که رنگ مربوط به گوشه ها** (edge corner) **بایستی برای پر کردن فضای اضافی مورد استفاده قرار گیرد. در حالی که ثابت** Shader.TileMode.MIRROR **اعلان می کند که تصویر مورد نظر بایستی قرینه سازی شده و** Shader.TileMode.REPEAT **نیز به اندروید اعلان می کند که عکسی را تکرار کند**.

ذخیره ی دائمی و ماندگارسازی داده های view

اغلب view **های متعارف اندروید قادرند داده های مربوط به وضعیت خود را نگه دارند. این داده ها سپس توسط سیستم به صورت دائمی ذخیره می شوند. سیستم اندروید جهت ذخیره ی داده های مربوط به وضعیت** view**متد** ()onSaveInstanceState **را صدا می زند و جهت ذخیره ی دائمی و بازگردانی این داده ها متد** onRestoreInstanceState(Parcable)**را فراخوانی می کند**.

برای ماندگار سازی داده ها، مرسوم است که کلاس View.BasedSavedState **را داخل** view **به صورت یک کلاس درونی** static **پیاده سازی نمایید**.

متد View.BasedSavedState **یک قرارداد شخصی برای نگه داری داده ها می باشد**.

اندروید یک view **را بر اساس** ID **آن در فایل** layout **میزبان پیدا کرده و سپس آبجکت** Bundle **را (که حاوی داده های مربوط به وضعیت** view **می باشد) به** view **ارسال می کند** .view **با داده های کپسوله شده در این آبجکت وضعیت خود را بازگردانی می نماید**.

شما می بایست اطلاعات UI **را دقیقا در همان وضعیتی که کاربر آن را ترک کرده، نگه دارید و بعد مجددا آن را بازگردانی نمایید. برای مثال موقعیت نوار پیمایش یا انتخاب کاربر را به حالت قبلی آن بازگردانید**.