Layouts

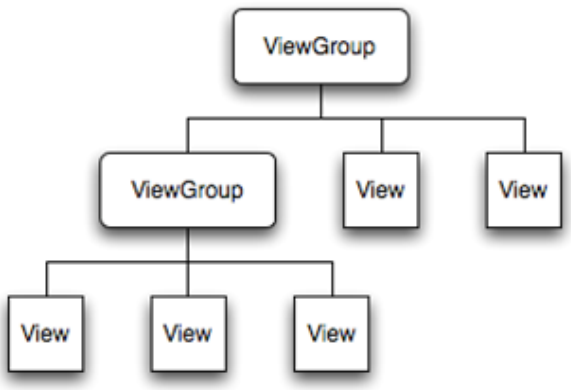
Organizacija grafičkih elemenata

Kao što je prethodno napomenuto,dokumenti pisani u XML jeziku, koji se koristi za definisanje formata podataka u Android-u, imaju hijerarhijsku strukturu stabla. Ovi dokumenti u okviru Andorid aplikacije definišu layout-e. Layout u Androidu definiše strukturu korisničkog interfejsa aplikacije. Prateći hijerarhijsku strukturu, svi elementi layout-a grade se hijerathijski koristeći View i ViewGroup objekte.

View i ViewGroup

ViewGroup je osnovna klasa za layout kontejnere koji sadrže View elemente kao svoju decu i odgovorni su sa njihov raspored na ekranu

View je osnovni element za pravljenje korisničke sprege. To je pravougaoni objekat na ekranu odgovoran za iscrtavanje sadržaja i obradu događaja. View komponente su organizovane u strukturu poput stabla i mogu biti definisane u XML deskriptoru ili u run-time-u.



Slika 1 – Hijerarhija View-a koji definiše UI layout

Deklarisanjem korisničkog interfejsa u XML datotekama odvajamo izgled aplikacije od programskog koda koji kontroliše ponašanje aplikacije. Takođe, omogućava korisniku da prilagodi izgled aplikacije za različite izglede, orijentacije i veličine ekrana uređaja za koje je aplikacija predviđena.

Prilikom prevođenja aplikacije, svaka XML datoteka se prevodi u View resurs. O tome kako se rukuje ovim resursima unutar programskog koda, biće reči u poglavlju Android Activity.

Svaki View I ViewGroup objekat podržava sebi svojstvene atribute. Neki su specifični u zavisnosti od View objekta, ali takođe i bivaju nasleđeni od strane bilo kog View objekta koji nasleđuje View klasu. Neki su, dakle, zajednički svim View objektima zbog toga što nasleđuju korensku View klasu. Primer takvog atributa je ID atribut. Ostali atributi se smatraju parametrima layout-a koji opisuju orijentaciju određenih objekata u okviru layout-a, kao i njihovu veličinu.

ID

Svaki View objekat može imate celobrojni identifikator sa kojim je povezan, kako bi se jedinstveno izdvojio u okviru View stabla. Prilikom prevođenja aplikacije, ovaj idetifikator se referencira kao celobrojna numerička vrednost, dok se prilikom dodeljivanja u okviru XML datoteke koriste String vrednosti. Sintaksa za upotrebu ovog atributa je:

```
android:id="@+id/moj_element"
```

Simbol (@) na početku ovog znakovnog niza označava da XML parser treba da parsira ostatak ID String-a i identifikuje ga kao ID resurs.

Simbol (+) označava da je ovo novi resurs koji se mora kreirati i uvrstiti među postojeće. Ne mora se koristiti (+) simbol, takođe se može navesti umesto njega i imenski prostor andorid paketa.

Svi resursi u okviru Andorid-a čuvaju se u R.java datoteci.

Layout parametri

Layout atributi sa nazivom layout_*nesto* definišu parametar layout-a za View objekat. Svaka ViewGroup klasa implementira ViewGropu.LayoutParams klasu u okviru koje se sadrže tipovi koji definišu veličinu i pozicionirajuće parametre svakog View-a naslednika. Svaki View mora definisati svoje vrednosti layout_width (širina) i layout_height (visina) parametara. Mnogi takođenude mogućnost definisanja margina i granica. Ovi parametri mogu s definisati preciznim vrednostima, što nije preporučljivo iz razloga što layout-i treba da izgledaju isto na svakom tipu i veličini ekrana uređaja, što nije zagarantovano ukoliko se ovde vrednosti definišu preciznim mernim jedinicama. Rešenje leži u upotrebi relativnih vrednosti:

- ***wrap_content*** - definiše da view ograniči svoju veličinu na dimenzije koje zauzima njegov sadržaj.
- ***match_parent*** - definiše da view prilagodi svoju veličinu veličini svoje roditeljske View grupe.

Najčešći layout-i

Svaka klasa koja nasleđuje ViewGropu klasu pruža jedinistveni način za prikaz view elemenata unutar nje. Neki od najčešće korišćenih layout tipova ugrađenih u Android platformu su:

- **Linear Layout –** layout koji svoje pripadnike organizuje u vertikalnu ili horizontalnu kolonu. Kreira se i traka za pomeranje vidljivog dela ekrana (engl.*scrollbar*) u zavisnosti od odnosa dužine prozora sa sadržajem i ekrana.
- **Relative Layout –** omogućava specificiranje pozicije pripadaućih objekata relativno u odnosu jedan na drugog, ili u zavisnosti od pozicije roditeljske grupe.

Ova dva layout-a su najzastupljenija, a pored njih su takođe prisutni i: ConstraintLayout, WebView,...

Najčešći elementi layout-a

Elemenata koji se grupišu unutar layout-a u okviru Android-a ima mnogo. Neki od najzastupljenijih su:

- TextView
- Button
- ListView
- EditText
- RadioButton

- CheckBox
- Itd…

Implementacija nekih od navedenih elemenata u okviru layout-a:

```
<Button
   android:id="@+id/button_id"
   android:layout_height="wrap_content"
   android:layout_width="wrap_content"
   android:text="@string/Dugme Start" />
```

Slika 2 - Dugme (engl. Button)

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
  <TextView
     android:id="@+id/text_view_id"
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:text="@string/Moj prvi text view" />
</LinearLayout>
```

Slika 3 - Prikaz teksta (engl. Text View)

```
<ListView
    android:id="@+id/list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Slika 4 - Lista (engl. ListView)

Activity

Aktivnost (engl. Activity) se naziva jedan prozor unutar aplikacije sa korisničkim interfejsom, koji je namenjen za interakciju korisnika sa aplikacijom. Potrebno je da svaka aplikacija ima bar jednu, glavnu aktivnost, koja se prikazuje prva pri pokretanju.

Kreiranje prazne aktivnosti: ***File->New->Activity->Empty Activity***

Prilikom kreiranja nove aktivnosti, omogućeno je korišćenje već predefinisanih (template) aktivnosti sa implementiranim grafičkim komponentama kao što su meniji, navigacioni dugmići, i sl.

Više detalja o postojećim template aktivnostima na linku :

https://developer.android.com/studio/projects/templates.html

Podaci o kreiranim aktivnostima se nalaze u AndroidManifest.xml fajlu.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="rtrk.pnrs.example" >
    <application
           . . .
           <!-- details about app -->
           . . .
           >
        <activity android:name=".MainActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
           <activity android:name=".SecondActivity"></activity>
    </application>
</manifest>
```

Svaka aktivnost nasleđuje klasu *Activity,* koja sadrži deklaracije callback metoda životnog cilklusa aktivnosti, čije objašnjenje sledi.

Stanja

Svaka aktivnost može da se nađe u jednom od stanja iz konačnog skupa u svom životnom ciklusu. Kako bi se na odgovarajući način moglo reagovati na prelazak u stanja, nasleđivanjem klase *Activity,* omogućeno je implementiranje callback metoda koje se pozivaju pri prelasku u određeno stanje. Sledi kratak opis svake od callback funkcija.

- **onCreate**

Ovo je jedina metoda koju je obavezno implementirati. Poziva se samo jednom, kada aktivnost pređe u stanje ”Created”, tj. kada je aktivnost inicijalizovana. Shodno tome, u njoj je potrebno implementirati inicijalizaciju komponenti. U njoj je potebno pozvati metodu *setContentView* kojoj se prosleđuje objekat klase View, koji je potrebno prikazati, ili .xml fajl sa definisanim grafičkim izgledom.

- **onStart**

Metoda onStart se aktivira odmah posle poziva onCreate metode. Po pozivu, aktivnost je vidljiva korisniku, ali još uvek ne može intereaguje sa njom. Po završetku izvršavanja, prelazi u stanje “Started” posle čega se poziva metoda onResume.

- **onResume**

Sada aktivnost prelazi u glavni plan i korisnik može da intereaguje sa njom. Aktivnost će ostati u ovom stanju sve dok se ne desi neki događaj koji bi skrenuo fokus sa nje, kao što je npr. gašenje ekrana, prelazak u drugu aktivnost, otvaranje dijaloga i sl. Tada aktivnost prelazi u stanje “Paused” i sistem poziva onPause callback metodu. Kada se aktivnost vrati u stanje “Resumed”, onResumed metoda će ponovo biti pozvana.

- **onPause**

Ova metoda se poziva kada se desi indikacija da korisnik napušta aktivnost ili kada ona više nije u prvom planu, kao npr. kada se prikazuje dijalog. U ovoj metodi je potrebno privremeno zaustaviti sve funkcije, a po povratku i pozivu onResume ponovo aktivirati. Aktivnost će biti u ovom stanju sve dok ne postane u potpunosti nevidljiva korisniku, kada će se pozvati metoda onStop i preći u stanje “Stopped”.

- **onStop**

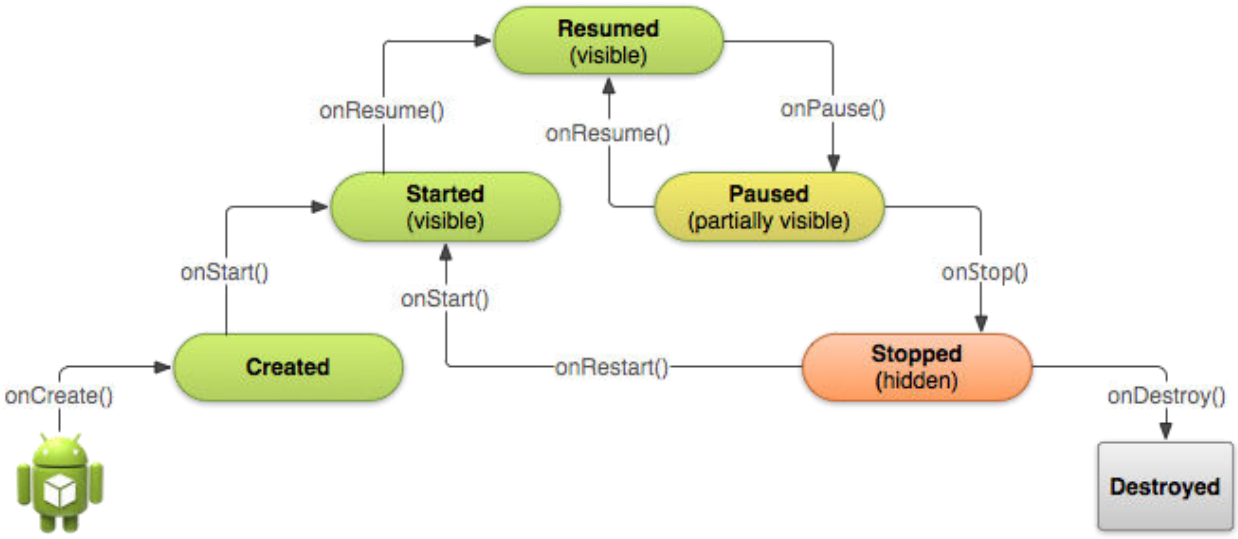
Kada aktivnost više nije vidljiva korisniku, aktivnost prelazi u stanje “Stopped” i sistem poziva metodu onStop. Ovo se najčešće dešava kada druga aktivnost stupi kao primarna na ekran. U ovoj metodi bi trebali da se oslobode svi resursi koji nisu potrebni dok je aktivnost u pozadini. Iz ovog stanja moguć je prelazak ili u ponovnu interakciju, ili aktivnost biva uništena i uklonjena iz memorije. Ukoliko se vrati u fokus, poziva se metoda onRestart.

- **onRestart**

Poziva se posle onStop metode ukoliko se korisnik ponovo vrati na nju. Ona je praćena pozivom onStart i onResume.

- **onDestroy**

Ova metoda se poziva kada je aktivnost uništena, tj. pre nego što pređe u stanje “Destroyed”. To se može dogoditi eksplicitno, pozivanjem funkcije *finish,* pritiskom dugmeta za povratak (back), ukoliko se desi rotacija ekrana, ili ako je operativnom sistemu potrebno da oslobodi resurse za drugu namenu. Sledeći put pri pokretanju aktivnosti, biće pozvana funkcija onCreate koja rekreira proces.



Intenti

Intent je klasa koja koordinira prenošenje poruka između komponenti aplikacije, koja sadrži opis pojedine akcije koja treba da se izvrši.

Ukoliko aplikacija ima više aktivnosti, potrebno je napraviti vezu između njih, i po potrebi preneti određene podatke. Što znači, da se iz jedne aktivnosti može pokrenuti (prikazati) druga aktivnost, na neki događaj (npr. klik dugmeta), i u tu svrhu se koristi klasa Intent.

Ova klasa ima i druge namene, pre svega, za pokretanje servisa, broadcast receiver-a, kao i prenošenje podataka između aktivnosti.

```
/*Example of starting new activity on button click */

    button = (Button) findViewById(R.id.button);
    button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                   Intent intent = new Intent(MainActivity.this,
            SecondActivity.class);
                   startActivity(intent);
        }
    });
```