

# Kreiranje korisničkog interfejsa primenom pogleda

*Doc. dr Vladimir Milićević*



# TEXTVIEW I EDITTEXT POGLEDI

*TextView pogled se koristi za prikazivanje teksta korisnicima aplikacije.*

TextView pogled se koristi za prikazivanje teksta korisnicima aplikacije i na drugim platformama se najčešće naziva labela. Radi se o osnovnom pogledu koji se skoro uvek koristi u Android aplikacijama. Prilikom kreiranja svakog novog Android projekta, Eclipse IDE razvojnim okruženjem, kreira se main.xml datoteka koja obavezno sadrži <TextView> element, a to je prikazano kodom sa sledeće slike. Ako je neophodno da se omogući korisnicima aplikacije da menjaju tekst, biće korišćen EditText klasa, izvedena iz TextView klase.

```
<RelativeLayout 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"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.metropolitan.hello.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Slika-1 Default TextView po kreiranju novog projekta

# OSTALI OSNOVNI POGLEDI

*Osnovni pogledi se najčešće kreiraju main.xml datotekom.*

Pored *TextView*, veoma često se koriste sledeći osnovni pogledi:

- *EditText* – omogućava ažuriranje teksta;
- *Button* – predstavlja dugme koje je moguće pritisnuti;
- *ImageButton* – dugme sa slikom;
- *CheckBox* – specijalni tip tastera sa dva stanja: selektovan (čekiran) i neselektovan (nečekiran);
- *ToggleButton* – dugme sa svetlosnim indikatorom za prikazivanje stanja selektovan/neselektovan;
- *RadioButton* – ima dva stanja selektovan i neselektovan;
- *RadioGroup* – Grupa radio dugmadi u kojoj samo jedan, u datom trenutku, može biti selektovan.

Sledećom main.xml datotekom definisani su osnovni pogledi.

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

    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/save"
        android:onClick="btnSaved_clicked"/>

    <Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open" />

    <ImageButton android:id="@+id/btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher" />

    <EditText android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Autosave" />

    <CheckBox android:id="@+id/star"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 1" />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Option 2" />
    </RadioGroup>

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Slika-2 Osnovni pogledi u main.xml

# UPRAVLJANJE OSNOVNIM POGLEDIMA

*Akcije koje su rezultat interakcije korisnika sa osnovnim pogledima definisane su JAVA klasom aktivnosti.*

U projektu, za demonstraciju upotrebe osnovnih pogleda, neophodno je modifikovati klasu aktivnosti tako da kreirane kontrole, main.xml datotekom, imaju konkretne zadatke. Navedeni zadaci definisani si sledećim JAVA programskim kodom.

```
package com.metropolitan.BasicViews1;
import android.app.Activity;
public class BasicViews1Activity extends Activity {

    public void btnSaved_clicked (View view) {
        DisplayToast("Kliknuli ste na dugme Save");
    }

    /** Poziva se kada se kreira aktivnost. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpen);
        btnOpen.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                DisplayToast("Kliknuli ste na dugme Open");
            }
        });

        //---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);
        checkBox.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((CheckBox)v).isChecked())
                    DisplayToast("CheckBox je čekiran");
                else
                    DisplayToast("CheckBox nije čekiran");
            }
        });

        //---RadioButton---
        RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()
        {
            public void onCheckedChanged(RadioGroup group, int checkedId) {
                RadioButton rbl = (RadioButton) findViewById(R.id.rdb1);
                if (rbl.isChecked()) {
                    DisplayToast("Option 1 je čekiran!");
                } else {
                    DisplayToast("Option 2 je čekiran!");
                }
            }
        });

        //---ToggleButton---
        ToggleButton toggleButton =
            (ToggleButton) findViewById(R.id.toggle1);
        toggleButton.setOnClickListener(new View.OnClickListener()
        {
            public void onClick(View v) {
                if (((ToggleButton)v).isChecked())
                    DisplayToast("Toggle button je uključen");
                else
                    DisplayToast("Toggle button je isključen");
            }
        });

        private void DisplayToast(String msg)
        {
            Toast.makeText(getApplicationContext(), msg,
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

Slika-3 Klasa aktivnosti za osnovne poglede

# OSNOVNI POGLEDI – NAČIN FUNKCIONISANJA

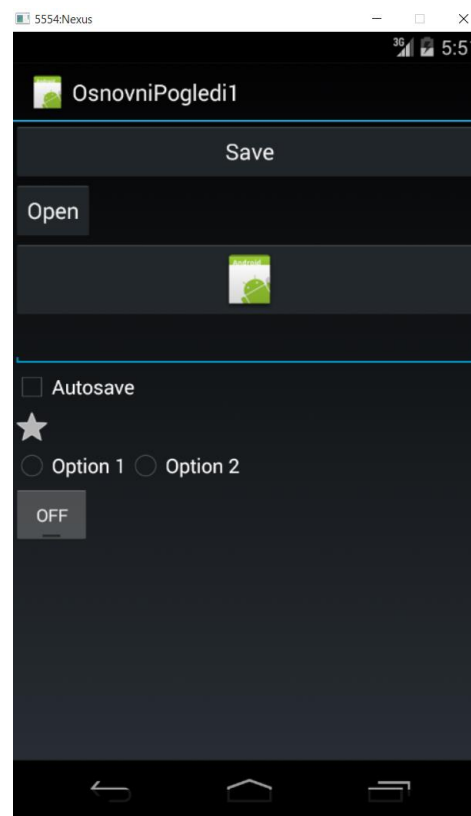
*Klikom na različite poglede moguće je primetiti kako se menjaju izgled i funkcija kontrole.*

Koristeći vertikalni `LinearLayout` raspored, pogledi su postavljeni jedan iznad drugog:

- Prvo dugme je definisano tako da je njegov element `layout_width` postavljen na `fill_parent`, a to znači da zauzima celu širinu ekrana;
- Drugo dugme ima vrednost `wrap_content` za naznačeni atribut, a to znači da je njegova širina definisana prostorom za prikazivanje sadržaja;
- `ImageButton` pomoću `src` atributa postavlja sliku;
- `EditText` pokazuje tekst polje za unos teksta;
- `CheckBox` definiše polje koje je moguće selektovati ili deselektovati. Upotrebom atributa `style` (videti `main.xml`) moguće je promeniti izgled polja (u zvezdicu u ovom primeru);
- Radio grupa sadrži dva dugmeta koji se međusobno isključuju;
- `ToggleButton` prikazuje taster koji ima dva stanja on/off.

Svaki pogled ima vlastiti identifikator definisan `id` atributom, kojeg je moguće koristiti pomoću metoda `View.findViewById()` i `Activity.findViewById()`.

Sledećom slikom prikazan je UI kreiran na navedeni način.



Slika-4 UI sa osnovnim pogledima

# UPRAVLJANJE DOGAĐAJIMA KOJI SE ODOSE NA POGLEDE

*Da bi upravljanje konkretnim pogledom bilo moguće, neophodno ga je programski locirati.*

Za svaki pogled, kreiran tokom `onCreate()` događaja, upravljanje je moguće ukoliko je pogled programski lociran. Navedeno je omogućeno prosleđivanjem `id` atributa metodi `findViewById()` koja pripada osnovnoj klasi `Activity`. Za dugme, koje je korišćeno u primeru, lociranje je rešeno na sledeći način:

```
Button btnOpen = (Button) findViewById(R.id.btnOpen);
```

Metodom `setOnClickListener()` omogućeno je dobijanje povratne informacije koja će biti prezentovana prilikom obraćanja pogledu:

```
btnOpen.setOnClickListener(new  
View.OnClickListener() {  
    public void onClick(View v) {  
        DisplayToast("Kliknuli ste na dugme  
Open");  
    }  
});
```

Stanje `CheckBox`-a proverava se pomoću argumenta za metodu `onClick()` kojim se, potom, poziva metoda `isChecked()` (sledeća slika).

Metoda `setOnCheckedChangeListener()` se koristi kada u radio grupi treba da se dobije informacija o promeni stanja radio dugmeta. Za svako dugme postoji metoda `isChecked()` kojom se proverava stanje kontrole (sledeća slika).

`ToggleButton` funkcioniše potpuno isto kao `CheckBox`.

```
CheckBox checkBox = (CheckBox) findViewById(R.id.chkAutosave);  
checkBox.setOnClickListener(new View.OnClickListener()  
{  
    public void onClick(View v) {  
        if (((CheckBox)v).isChecked())  
            DisplayToast("CheckBox je čekiran");  
        else  
            DisplayToast("CheckBox nije čekiran");  
    }  
});  
  
RadioGroup radioGroup = (RadioGroup) findViewById(R.id.rdBgp1);  
radioGroup.setOnCheckedChangeListener(new OnCheckedChangeListener()  
{  
    public void onCheckedChanged(RadioGroup group, int checkedId) {  
        RadioButton rb1 = (RadioButton) findViewById(R.id.rdB1);  
        if (rb1.isChecked()) {  
            DisplayToast("Option 1 je čekiran!");  
        } else {  
            DisplayToast("Option 2 je čekiran!");  
        }  
    }  
});
```

Slika-5 Upravljanje događajima pogleda

# PROGRESSBAR - FUNKCIONISANJE

*Inicijalno, ProgressBar je realizovan kao ciklična animacija.*

Kontrola *ProgressBar* je veoma korisna za zadatke koji nemaju specifično vreme izvršavanja, poput: send i response komunikacije sa web serverom. Postavljanjem xml taga `<ProgressBar>` u main.xml datoteku, biće obezbeđeno prikazivanje ikone koja se okreće. Obaveza programera je da obezbedi zaustavljanje kontrole kada se završi obavljanje datog zadatka.

*ProgressBar* kontrola, u ovom primeru, prati pozadinsku nit za simuliranje zadatka koji dugo traje. Otuda je neophodno imlementiranje klase *Thread* i *Runnable* objekat. Metodom `run()` započinje izvršavanje niti, koja u konkretnom primeru izvršava metodu, simbolično nazvanu, `doSomeWork()`. Po završetku zadatka, upotrebljen je *Handler* objekat za slanje poruke do niti za prekidanje *ProgressBar*-a. Za navedeno videti priloženi kod.

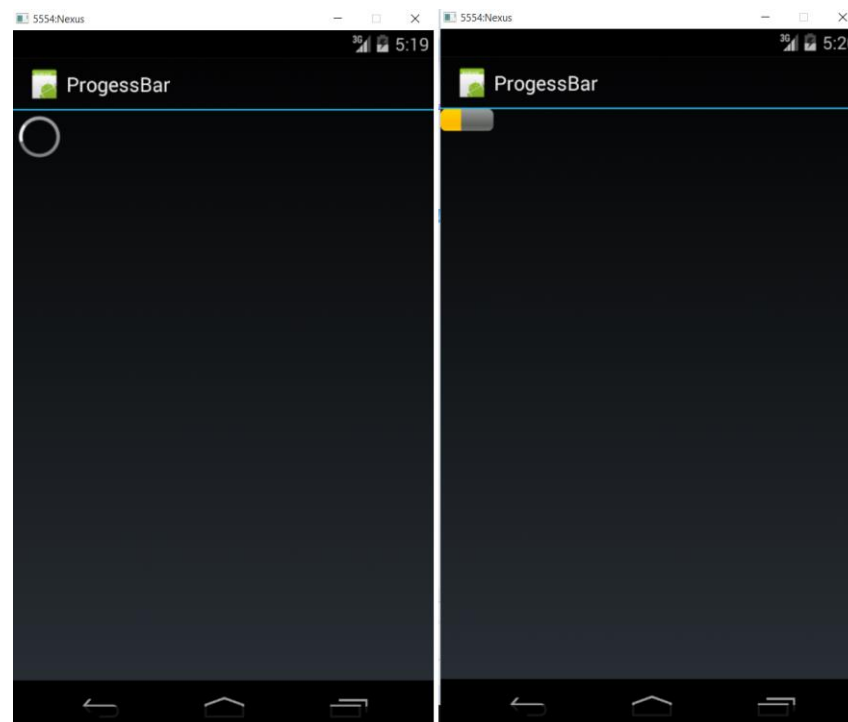
Inicijalno, *ProgressBar* je realizovan kao ciklična animacija.

Međutim, upotrebom instrukcije:

```
style="@android:style/Widget.ProgressBar.Horizontal,,
```

Može biti prikazan kao horizontalna animacija (sledeća slika).

Slikom je prikazana kontrola kao ciklična i horizontalna animacija.



Slika-8 ProgresBar kao različite animacije



# TIMEPICKER POGLED

## *TimePicker pogled omogućava korisnicima selektovanje vremena.*

Izbor datuma i vremena su akcije koje se veoma često izvode Android aplikacijama. Android obezbeđuje dva alata kojima su ove funkcionalnosti dostupne, a to su **TimePicker** i **DatePicker**.

*TimePicker* pogled omogućava korisnicima da izaberu vreme, i to, u dva režima 24h ili AM/PM. Korišćenje *TimePicker* pogleda biće demonstrirano sledećim primerom.

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

<Button android:id="@+id/btnSet"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="I am all set!"
    android:onClick="onClick" />

<DatePicker android:id="@+id/datePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TimePicker android:id="@+id/timePicker"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>
```

Slika-1 TimePicker - main.xml

Glavnu datoteku aktivnosti neophodno je modifikovati na sledeći način, pri čemu će biti uključen kod i za kontrolu *DatePicker*.

```
package com.metropolita.Picker;
import java.text.SimpleDateFormat;
public class PickerActivity extends Activity {
    TimePicker timePicker;
    DatePicker datePicker;
    int hour, minute;
    int yr, month, day;
    static final int TIME_DIALOG_ID = 0;
    static final int DATE_DIALOG_ID = 1;
    /** Poziva se kada se kreira aktivnost */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        timePicker = (TimePicker) findViewById(R.id.timePicker);
        datePicker = (DatePicker) findViewById(R.id.datePicker);

        // showDialog(TIME_DIALOG_ID);
        datePicker = (DatePicker) findViewById(R.id.datePicker);

        //---uzima tekući datum---
        Calendar today = Calendar.getInstance();
        yr = today.get(Calendar.YEAR);
        month = today.get(Calendar.MONTH);
        day = today.get(Calendar.DAY_OF_MONTH);

        showDialog(DATE_DIALOG_ID);
    }
    @Override
    protected Dialog onCreateDialog(int id)
    {
        switch (id) {
            case TIME_DIALOG_ID:
                return new TimePickerDialog(
                    this, mTimeSetListener, hour, minute, false);
            case DATE_DIALOG_ID:
                return new DatePickerDialog(
                    this, mDateSetListener, yr, month, day);
        }
        return null;
    }

    private DatePickerDialog.OnDateSetListener mDateSetListener =
        new DatePickerDialog.OnDateSetListener()
    {
        public void onDateSet(
            DatePicker view, int year, int monthOfYear, int dayOfMonth)
        {
            yr = year;
            month = monthOfYear;
            day = dayOfMonth;
            Toast.makeText(getBaseContext(),
                "Izabrali ste : " + (month + 1) +
                "/" + day + "/" + yr,
                Toast.LENGTH_SHORT).show();
        }
    };
    private TimePickerDialog.OnTimeSetListener mTimeSetListener =
        new TimePickerDialog.OnTimeSetListener()
    {
        public void onTimeSet(
            TimePicker view, int hourOfDay, int minuteOfHour)
        {
            hour = hourOfDay;
            minute = minuteOfHour;

            SimpleDateFormat timeFormat = new SimpleDateFormat("hh:mm aa");
            Date date = new Date(0,0,0, hour, minute);
            String strDate = timeFormat.format(date);

            Toast.makeText(getBaseContext(),
                "Izabrali ste " + strDate,
                Toast.LENGTH_SHORT).show();
        }
    };
    public void onClick(View view) {
        Toast.makeText(getBaseContext(),
            "Izabrani datum:" + (datePicker.getMonth() + 1) +
            "/" + datePicker.getDayOfMonth() +
            "/" + datePicker.getYear() + "\n" +
            "Izabrano vreme:" + timePicker.getCurrentHour() +
            ":" + timePicker.getCurrentMinute(),
            Toast.LENGTH_SHORT).show();
    }
}
```

Slika-2 TimePicker - JAVA klasa aktivnosti



# TIMEPICKER - FUNKCIONISANJE

*TimePicker u standardnom Android UI omogućava korisniku da podesi vreme.*

*TimePicker* koristi standardni korisnički interfejs u kome je korisniku aplikacije omogućeno da podesi vreme. Po osnovnim podešavanjima, vreme se prikazuje u formatu AM/FM. Ukoliko se želi prikaz vremena u 24h formatu, neophodno je koristiti metodu *setIs24HourView()* (videti metodu *onCreate()* priloženog koda). Da bi vreme bilo programski prikazano, neophodno je upotrebiti metode *getCurrentHour()* i *getCurrentMinute()*, na sledeći način:

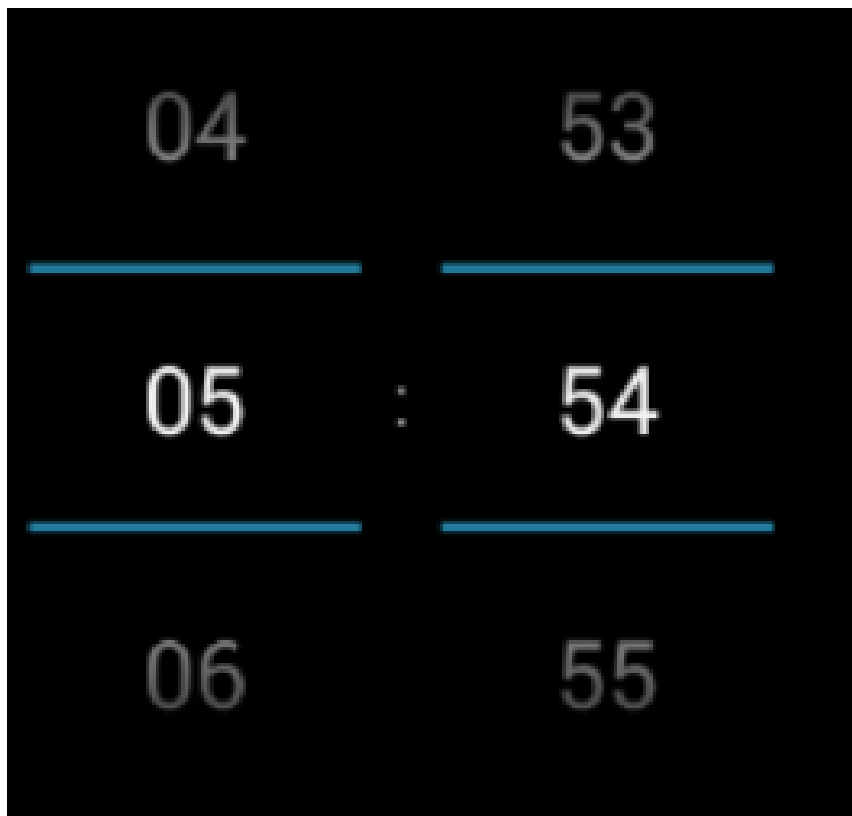
```
Toast.makeText(getApplicationContext(),  
"Izabrano vreme:" + timePicker.getCurrentHour() +  
":" + timePicker.getCurrentMinute(),
```

# PAKOVANJE TIMEPICKER POGLEDA U OKVIR DIJALOGA

*Prosleđivanjem odgovarajućeg indikatora metodi `showDialog`, `TimePicker` pogled se smešta u okvir dijaloga.*

Do sada je poznato da poziv dijaloga okvira inicira izvršavanje metode `showDialog()`. U konkretnom slučaju (videti priloženi kod) ovoj metodi je prosleđen atribut `TIME_DIALOG_ID` kojim je identifikovan izvor dijaloga okvira. Izvršavanjem metode `showDialog()`, za prikazivanje okvira, izvršava se i metoda `onCreateDialog()` u okviru koje se kreira instanca klase `TimePickerDialog`. Navedena instanca će preuzeti informacije od značaja, a to su u konkretnom slučaju sati i minuti. Na kraju, kao što je u kodu naznačeno, podaci će biti prezentovani u 24h formatu.

Sledećom slikom je prikazan okvir dijaloga za `TimePicker`.



Slika-3 TimePicker - okvir dijaloga

# DATEPICKER POGLED

*DatePicker pogled omogućava korisnicima selekciju datuma.*

Priloženim kodom definisan je `DatePicker` pogled koji je po načinu korišćenja veoma sličan `TimePicker` pogledu. Primenom ovog pogleda, u određenoj aktivnosti, korisnicima je omogućeno da izaberu određeni datum. Takođe, `main.xml` datotekom definiše se `DatePicker` pogled odgovarajućim XML tagom (sledeća slika).

```
<DatePicker android:id="@+id/datePicker"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

Slika-4 DatePicker XML tag

# DATEPICKER POGLED - FUNKCIONISANJE

*DatePicker pogled koristi specifične metode za prikazivanje podataka o datumu.*

Na sličan način kao *TimePicker*, *DatePicker* pogled koristi izvesne metode za prikazivanje podataka od značaja: dan, mesec i godina. Za prikazivanje ovih podataka odgovorne su metode *getDayOfMonth()*, *getMonth()* i *getYear()*, respektivno. Trebalo bi napomenuti da metoda *getMonth()* vraća vrednost 0 za mesec januar. Otuda, prilikom upotrebe ove metode vrši se inkrementacija rezultata da bi on imao valjanu vrednost. Primena ovih metoda je ugrađena u priloženi kod, a ovde će biti prezentovan samo konkretan deo poziva metoda.

```
"Izabrani datum:" + (datePicker.getMonth() + 1) +  
"/" + datePicker.getDayOfMonth() +
```

# PAKOVANJE DATEPICKER POGLEDA U OKVIR DIJALOGA

*Kao i `TimePicker`, `DatePicker` može biti prikazan u okviru za dijalog.*

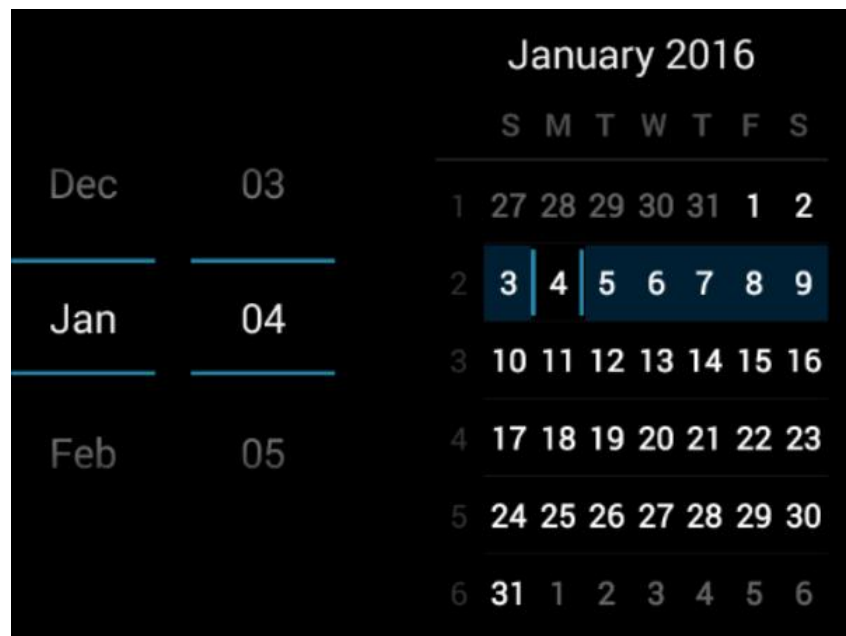
Kao i `TimePicker` pogled, `DatePicker` pogled može biti prikazan u okviru za dijalog. Identifikator za `DatePicker` definisan je kao `DATE_DIALOG_ID` koji se predaje metodi `showDialog()`. Po ovome, `DatePicker` i `TimePicker` funkcionišu na isti način. Nakon definisanja datuma, poziva se metoda `onSetDate()` koja kao rezultat vraća datum koji je korisnik izabrao (videti priloženi kod).

Trebalo bi napomenuti da je neophodno inicijalizovati promenljive za dan, mesec i godinu pre prikazivanja okvira za dijalog (vidi sliku). U suprotnom javiće se izuzetak, tokom izvršavanja programa, zbog upotrebe ilegalnog argumenta.

```
yr = year;  
month = monthOfYear;  
day = dayOfMonth;  
Toast.makeText(getApplicationContext(),  
    "Izabrali ste : " + (month + 1) +  
    "/" + day + "/" + year,  
    Toast.LENGTH_SHORT).show();
```

Slika-5 Varijable datuma

Sledećom slikom prikazan je okvir dijaloga `DatePicker` pogleda.



Slika-14 `DatePicker` okvir dijaloga

# LISTVIEW POGLED

*Prikazivanje liste stavki u vertikalnoj skrol listi omogućeno je ListView pogledom.*

List pogledi omogućavaju prikazivanje dugačkih listi. U Android operativnom sistemu podrška prikazivanju dugih listi realizovana je primenom pogleda **ListView** i **SpinnerView**. Prikazivanje liste stavki u vertikalnoj skrol listi omogućeno je **ListView** pogledom.

Sledećim primerom biće prikazana upotreba **ListView** pogleda.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
<Button android:id="@+id/btn"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Prikaži izabrane stavke"
    android:onClick="onClick"/>
<ListView
    android:id="@+id/android:list"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
</LinearLayout>
```

Slika-1 ListView - main.xml

Klasa aktivnosti primera data je kodom sa slike.

```
package com.metropolitan.ListView;
import android.app.ListActivity;
public class ListViewActivity extends ListActivity {
    String[] predmeti;
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ListView listView = getListView();
        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        listView.setTextFilterEnabled(true);

        predmeti =
            getResources().getStringArray(R.array.predmeti_array);

        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_checked, predmeti));
    }
    public void onItemClick(
        ListView parent, View v, int position, long id)
    {
        Toast.makeText(this,
            "Izabrali ste " + predmeti[position],
            Toast.LENGTH_SHORT).show();
    }
    public void onClick(View view) {
        ListView listView = getListView();
        String itemsSelected = "Izabrana stavka: \n";
        for (int i=0; i<listView.getCount(); i++) {
            if (listView.isItemChecked(i)) {
                itemsSelected += listView.getItemAtPosition(i) + "\n";
            }
        }
        Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();
    }
}
```

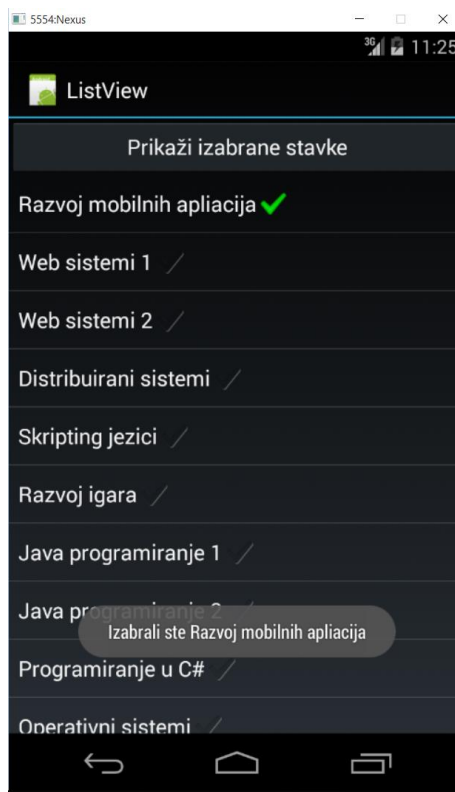
Slika-2 ListView - JAVA klasa aktivnosti

# LISTVIEW POGLED - FUNKCIONISANJE

*Klasa za implementaciju ListView pogleda nasleđuje klasu ListActivity.*

Kada se pogleda priloženi kod primera moguće je uočiti da klasa aktivnosti, koja je kreirana za prikaz liste stavki, nasleđuje klasu *ListActivity* koja je, pak, potklasa bazne klase *Activity*. Takođe, datoteku *main.xml* nije potrebno modifikovati da bi *ListView* pogled bio uključen. Klasa koja nasleđuje *ListActivity* već sadrži *ListView*. Zbog toga, u *onCreate()* metodi nije potrebno pozvati metodu *setContentView()* sa ciljem učitavanja korisničkog interfejsa iz datoteke *main.xml*. U *onCreate()* metodi se koristi *setListAdapter()* za popunjavanje ekrana aktivnošću koja odgovara *ListView* pogledu. Komponenta *ArrayAdapter* upravlja nizom stringova prezentovanih *ListView* pogledom. Ako se pogleda priloženi kod, *ListView* pogled je definisan da bude prikazan u jednostavnom *simple\_list\_item\_checked* režimu.

Na kraju, metoda *onListItemClick()* se inicira uvek kada korisnik klikne na neku stavku iz liste. Sledećom slikom prikazan je korisnički interfejs koji odgovara tekućem primeru.



Slika-3 ListView pogled



# PODEŠAVANJE LISTVIEW POGLEDA

*ListView pogled podržava različite prikaze koji se mogu posebno podešavati.*

ListView pogled podržava različite prikaze koji se mogu posebno podešavati. U konkretnom primeru definisano je da iz liste, u svakom trenutku izvršavanja programa, može biti izabrano više stavki. Ovo je moguće prilagoditi na dva načina: nije dozvoljeno selektovanje iz liste ili moguće je selektovati samo jednu stavku liste. Navedeno je prikazano sledećim kodom:

```
public class BasicViewsActivity extends ListActivity {
    String[] predmeti;
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ListView listView = getListView();
        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
        listView.setTextFilterEnabled(true);

        predmeti =
            getResources().getStringArray(R.array.predmeti_array);

        setListAdapter(new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_checked, predmeti));
    }
}
```

Slika-4 Podešavanje ListView pogleda

Lista ne dozvoljava izbor ukoliko se u metodi `setChoiceMode()` kao argument javi konstanta `CHOICE_MODE_NONE`, a iz liste je moguće izabrati samo jednu stavku ukoliko je ovoj metodi pridružena konstanta `CHOICE_MODE_SINGLE`.

Da bi navedenu metodu bilo moguće koristiti, neophodno je kreirati objekat klase `ListView`, za čije je učitavanje neophodno pozvati metodu `getListView()`.

Još jedan koristan element, koji je u primeru bio korišćen, jeste podrška za filtriranje. Nakon omogućavanja metode `setTextFilterEnabled()`, otvara se mogućnost za unos teksta pomoću tastature, a `ListView` prikaz će automatski biti modifikovan da se filtriraju samo one stavke koje odgovaraju unetim podacima: `ListView.setTextFilterEnabled(true);`

# ČUVANJE STAVKI U STRING.XML DATOTEKI

*Stavke liste moguće je čuvati izvan klase aktivnosti aplikacije.*

U jednostavnim aplikacijama moguće je čuvati podatke kao niz u JAVA klasi aktivnosti aplikacije. Međutim, u realnim uslovima, poželjnije je čuvati stavke u bazama podataka ili, pak, u datotekama. U konkretnom primeru, odabrana je datoteka strings.xml da sačuva članove liste do njihovog poziva. Datoteka je data sledećim kodom:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">ListView demonstracija!</string>
  <string name="app_name">ListView</string>
  <string-array name="predmeti_array">
    <item>Razvoj mobilnih aplikacija</item>
    <item>Web sistemi 1</item>
    <item>Web sistemi 2</item>
    <item>Distribuirani sistemi</item>
    <item>Skripting jezici</item>
    <item>Razvoj igara</item>
    <item>Java programiranje 1</item>
    <item>Java programiranje 2</item>
    <item>Programiranje u C#</item>
    <item>Operativni sistemi</item>
    <item>Matematika</item>
  </string-array>
</resources>
```

Slika-5 Čuvanje stavki liste u datoteci

Budući da su nazivi predmeta sačuvani u string.xml datoteci, njih je moguće učitati, na veoma jednostavan način, upotrebom metode `getResources()` u klasi aktivnosti aplikacije.

# PROVERA SELEKTOVANIH STAVKI

*Metoda `isItemChecked()` ispituje da li je stavka izabrana ili ne.*

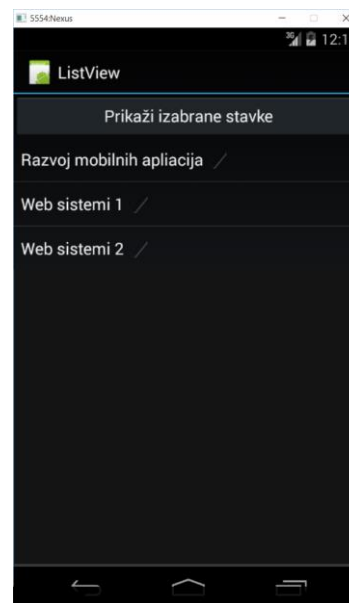
U ovom delu lekcije je istaknuto kako se prikazuje *ListView* pogled koji zauzima celu površinu za prikazivanje određene aktivnosti i nije bilo potrebe za dodavanjem `<ListView>` elementa u `main.xml`. Da bi prostor predviđen za celokupnu aktivnost bio delimično ispunjen, neophodno je dodati `<ListView>` element u `main.xml` datoteku na način prikazan priloženim programskim kodom za ovaj primer. Ovaj element mora da bude snabdeven i *id* atributom čija je vrednost: `@+id/android:list`. To znači da je u ovom slučaju neophodno učitati metodu `setContentViews()` (videti priloženi kod klase aktivnosti) da bi interfejs iz datoteke `main.xml` bio učitani.

Metodom `isItemChecked()` proverava se da li stavka u listi izabrana, ili ne, na sledeći način:

```
ListView listView = getListView();
String itemsSelected = "Izabrana stavka: \n";
for (int i=0; i<listView.getCount(); i++) {
    if (listView.isItemChecked(i)) {
        itemsSelected += listView.getItemAtPosition(i) + "\n";
    }
}
Toast.makeText(this, itemsSelected, Toast.LENGTH_LONG).show();
```

Slika-6 Provera selekcije stavki

U slučaju učitavanja liste iz niza i *ListView* pogleda, definisanih klasom aktivnosti aplikacije, bez primene metode `setContentViews()`, pogled bi zauzeo celokupnu površinu namenjenu za aktivnost i UI aplikacije bi izgledao na sledeći način.



Slika-7 Učitavanje *ListView* iz klase

# SPINNERVIEW POGLED

*SpinnerView pogled omogućava prikazivanje jedne po jedne stavke iz liste.*

Ukoliko je neophodno da se pored liste stavki, u odgovarajućoj aktivnosti, prikažu i drugi pogledi, a da se ne zauzme cela površina ekrana kao kod *ListView* pogleda, trebalo bi koristiti *SpinnerView* prikaz. Ovaj prikaz omogućava prikazivanje jedne po jedne stavke iz liste.

Prethodni primer biće modifikovan. Datoteka *main.xml* imaće sledeći kod.

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

<Spinner
    android:id="@+id/spinner1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawSelectorOnTop="true" />

</LinearLayout>
```

Slika-8 Spinner primer - main.xml

Datoteka *SpinnerActivity.java* imaće sledeći kod.

```
package com.metropolita.Spinner;

import android.app.Activity;

public class SpinnerActivity extends Activity {
    String[] predmeti;

    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        predmeti =
            getResources().getStringArray(R.array.predmeti_array);
        Spinner s1 = (Spinner) findViewById(R.id.spinner1);

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_single_choice, predmeti);

        s1.setAdapter(adapter);
        s1.setOnItemClickListener(new OnItemSelectedListener()
        {
            @Override
            public void onItemSelected(AdapterView<?> arg0,
                View arg1, int arg2, long arg3)
            {
                int index = arg0.getSelectedItemPosition();
                Toast.makeText(getApplicationContext(),
                    "Izabrali ste stavku : " + predmeti[index],
                    Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {}
        });
    }
}
```

Slika-9 Spinner primer - java klasa aktivnosti

# SPINNERVIEW POGLED - FUNKCIONISANJE

*Za SpinnerView neophodno je implementirati njegovu metodu `onNothingSelected()`.*

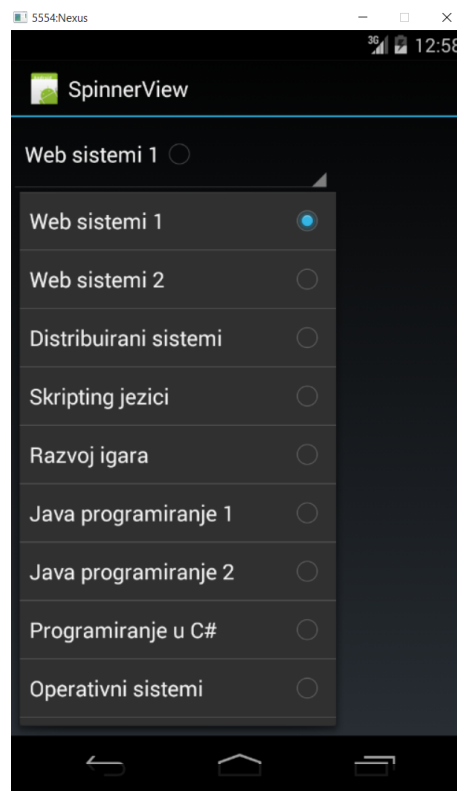
Tekući primer funkcioniše veoma slično prethodnom. Za početak je neohodno implementirati metodu `onNothingSelected()`. Ova metoda se izvršava kada korisnik pritisne taster *Back* na mobilnom uređaju i na taj način prekida dalje prikazivanje liste.

Umesto stavki za *ArrayAdapter*, kojima se prikazuje jednostavna lista, moguće je prikazati elemente korišćenjem radio tastera. Da bi to bilo moguće neophodno je modifikovati drugi parametar u konstruktoru *ArrayAdapter* klase, kao što je to urađeno u priloženom kodu primera:

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_single_choice, predmeti);
```

Slika-10 Dodavanje argumenta konstruktoru

Klikom na F11, program se prevodi i pokreće emulatorom i daje korisnički interfejs kao na sledećoj slici.



Slika-11 SpinnerView pogled

# LISTFRAGMENT KLASA

*ListFragment* klasa definiše fragment koji sadrži *ListView* pogled.

Kao što je naznačeno, u jednoj od prethodnih lekcija, fragmenti su mini-aktivnosti koje imaju sopstvene životne cikluse. Da bi fragment bio kreiran, on mora da bude podržan klasom koja nasleđuje baznu klasu *Fragment*. Pored osnovne klase omogućeno je i nasleđivanje izvesnih njenih potklasa sa ciljem kreiranja specijalizovanih fragmenata. Potklase klase *Fragment* su: *ListFragment*, *DialogFragment* i *PreferenceFragment*.

*ListFragment* klasa definiše fragment koji sadrži *ListView* pogled. *ListFragment* je veoma koristan alat jer omogućava prikazivanje liste stavki na ekranu zajedno sa ostalim elementima korisničkog interfejsa, a to može biti i još neka lista stavki. Da bi fragment za prikazivanje liste bio kreiran, neophodno je da ključna klasa nasledi klasu *ListFragment*.

# RES/LAYOUT DATOTEKE ZA DEMONSTRACIJU LISTFRAGMENT

*Za svaki fragment je neophodno kreirati odgovarajuću XML datoteku koja definiše njegov sadržaj.*

Kao i svaka Android aplikacija, aplikacija koja obrađuje fragmente mora da sadrži main.xml datoteku. Neka je u ovom slučaju ona data sledećim kodom.

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

<fragment
    android:name="net.learn2develop.ListFragmentExample.Fragment1"
    android:id="@+id/fragment1"
    android:layout_weight="0.5"
    android:layout_width="0dp"
    android:layout_height="200dp" />

<fragment
    android:name="net.learn2develop.ListFragmentExample.Fragment1"
    android:id="@+id/fragment2"
    android:layout_weight="0.5"
    android:layout_width="0dp"
    android:layout_height="300dp" />

</LinearLayout>
```

Slika-1 main.xml datoteka ListFragmet primera

Za fragment, koji je dobio naziv *Fragment1*, kreirana je XML datoteka naziva *Fragment1.xml* u kojoj se čuva definicija njegovog sadržaja. Datoteka je data sledećim XML kodom:

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

    <ListView
        android:id="@id/android:list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

</LinearLayout>
```

Slika-2 Fragment1.xml datoteka ListFragmet primera



# JAVA DATOTEKE APLIKACIJE ZA DEMONSTRACIJU LISTFRAGMENT

*Pored glavne JAVA klase, koja je zadužena za aktivnosti, neophodno je kreirati i JAVA klasu koja nasleđuje klasu ListFragment.*

Svaka Android aplikacija mora da sadrži glavnu JAVA klasu aktivnosti. Zadatak ove klase je, u velikom broju slučajeva, da obezbedi izvršavanje metode `setContentView()` kojom se učitava korisnički interfejs definisan u main.xml datoteci.

U ovom primeru, klasa aktivnosti će imati isključivo navedeni zadatak, a to je prikazano sledećim kodom.

```
package net.learn2develop.ListFragmentExample;

import android.app.Activity;

public class ListFragmentExampleActivity extends Activity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Slika-3 JAVA klasa aktivnosti primera

Da bi fragment za prikazivanje listi bio implementiran, neophodno je kreirati njegovu klasu. Kao što je napomenuto, on mora da nasledi osnovnu `ListFragment` klasu, a to je pokazano sledećim kodom.

```
package net.learn2develop.ListFragmentExample;
import android.app.ListFragment;
public class Fragment1 extends ListFragment {
    String[] predmeti = {
        "Razvoj mobilnih aplikacija",
        "Web sistemi 1",
        "Web sistemi 2",
        "Distribuirani sistemi",
        "Skripting jezici",
        "Razvoj igara",
        "Java programiranje 1",
        "Java programiranje 2",
        "Programiranje u C#",
        "Operativni sistemi",
        "Matematika"
    };
    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment1, container, false);
    }
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setListAdapter(new ArrayAdapter<String>(getActivity(),
            android.R.layout.simple_list_item_1, predmeti));
    }
    public void onItemClick(ListView parent, View v,
        int position, long id)
    {
        Toast.makeText(getActivity(),
            "Izabrali ste " + predmeti[position],
            Toast.LENGTH_SHORT).show();
    }
}
```

Slika-4 Klasa fragmenta za prikazivanje listi

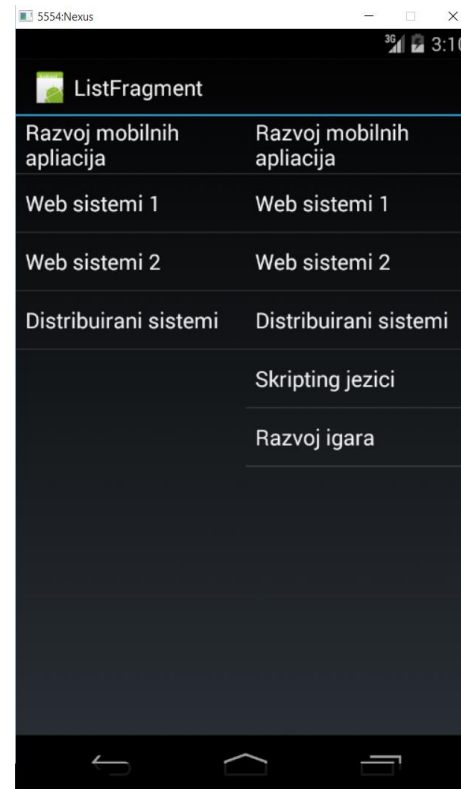
# APLIKACIJA ZA DEMONSTRACIJU LISTFRAGMENT - FUNKCIONISANJE

*Aplikacija pokazuje dva fragmenta za prikaz listi, postavljena jedan pored drugog.*

Kreiranje aplikacije, čiji je kod priložen, teče u sledećim koracima:

- Kreirana je XML datoteka za fragment koja sadrži <ListView> element;
- Kreirana je JAVA klasa koja nasleđuje klasu *ListFragment*;
- U ovoj klasi je kreiran niz predmeta (polje) koji će biti upisan u listu;
- Metoda *onCreate()* koristi metodu *setListAdapter()* čiji je zadatak punjenje liste sadržajem polja; *ArrayAdapter* upravlja nizom stringova koji će biti prikazani u *ListView* rasporedu jednostavnim režimom *simple\_list\_item\_1* (videti kod fragment klase).
- Metoda *onListItemClick()* poziva se prilikom svakog klika na stavku u *ListView* prikazu;
- Na kraju, datotekom *main.xml* je definisan interfejs koji poseduje dva fragmenta za prikazivanje listi. Za svaki fragment određena je drugačija visina prostora koji će zauzeti (videti kod *main.xml*).

Klikom na F11, primer se prevodi i pokreće emulatorom. Rezultat pokretanja aplikacije prikazan je sledećom slikom.



Slika-5 Dva List fragmenta

# DIALOGFRAGMENT KLASA

*Fragmenti za dijalog su veoma korisni kada bi trebalo dobiti odgovor korisnika pre nego što se nastavi izvršavanje Android aplikacije.*

Postoji još jedan tip fragmenata koji je moguće kreirati i čiji je zadatak da od korisnika dobije neki odgovor pre nego što se nastavi izvršavanje Android aplikacije. Ovakvi fragmeti su izvedeni iz klase *DialogFragment* i nazivaju se fragmentima za prikazivanje dijaloga.

Takođe, uvodi se primer za demonstraciju primene fragmenata za prikazivanje dijaloga. Sledeća slika daje kod datoteke main.xml.

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />

</LinearLayout>
```

Slika-6 DialogFragment primer - main.xml

# DIALOGFRAGMENT KLASA – JAVA KLASA PRIMERA

*Da bi fragment za prikazivanje dijaloga bio kreiran, neophodno je da ključna klasa nasledi klasu `DialogFragment`.*

Svaki fragment da bi bio implementiran, mora da poseduje vlastitu klasu. Klasa fragmenta za prikazivanje dijaloga mora da nasleđuje baznu klasu `DialogFragment` na način prikazan sledećim kodom.

```
package net.learn2develop.DialogFragmentExample;
import android.app.AlertDialog;
public class Fragment1 extends DialogFragment {
    static Fragment1 newInstance(String title) {
        Fragment1 fragment = new Fragment1();
        Bundle args = new Bundle();
        args.putString("Naslov", title);
        fragment.setArguments(args);
        return fragment;
    }
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        String title = getArguments().getString("Naslov");
        return new AlertDialog.Builder(getActivity())
            .setIcon(R.drawable.ic_launcher)
            .setTitle(title)
            .setPositiveButton("Da",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                        int whichButton) {
                        ((DialogFragmentExampleActivity)
                            getActivity()).doPositiveClick();
                    }
                })
            .setNegativeButton("Ne",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
                        int whichButton) {
                        ((DialogFragmentExampleActivity)
                            getActivity()).doNegativeClick();
                    }
                })
            .create();
    }
}
```

Slika-7 Klasa dijalog fragmenta

Glavna klasa aktivnosti aplikacije data je sledećim programskim kodom.

```
package net.learn2develop.DialogFragmentExample;

import android.app.Activity;

public class DialogFragmentExampleActivity extends Activity {
    /** Poziva se kada se kreira aktivnost. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Fragment1 dialogFragment = Fragment1.newInstance(
            "Da li želite da nastavite?");
        dialogFragment.show(getFragmentManager(), "dialog");
    }

    public void doPositiveClick() {
        //---Korisnik je kliknuo Da---
        Log.d("DialogFragment - primer", "Korisnik je kliknuo Da");
    }

    public void doNegativeClick() {
        //---Korisnik je kliknuo Ne---
        Log.d("DialogFragment - primer", "Korisnik je kliknuo Ne");
    }
}
```

Slika-8 Klasa aktivnosti aplikacije

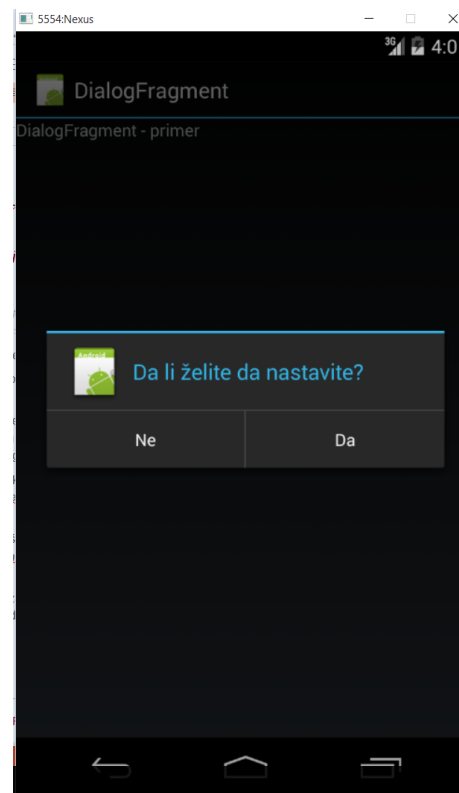
# FUNKCIONISANJE DIALOGFRAGMENT APLIKACIJA

## *Prozorom sa porukom i tasterima za reakciju, aplikacija čeka na odgovor korisnika*

Funkcionisanje aplikacije, koja implementira fragment sa dijalogom, podrazumeva da su ispunjeni sledeći uslovi:

- Kreirana fragment klasa koja nasleđuje klasu *Fragment*;
- Kreiran okvir za dijalog koji prikazuje poruku i odgovarajuće tastere;
- Metodom *newInstance()* omogućeno je kreiranje nove instance fragmenta i prihvatanje string argumenta koji će biti prikazan kao poruka u okviru za dijalog (videti priloženi kod);
- Postojanje *onCreateDialog()* metode, koja se izvršava nakon *onCreate()* metode, a pre *onCreateView()* metode (videti priloženi kod);
- Kreirana instanca fragmenta koja izvršava metodu *show()*:  
`dialogFragment.show(getFragmentManager(), "dialog");`
- Na kraju, neophodno je implementirati metode *doPositiveClick()* i *doNegativeClick()* za rukovanje aktivnostima koje su rezultat klika na dugmad *Da* i *Ne*, respektivno (videti priloženi kod).

Kada se ispoštuju ovi koraci i kreira priloženi kod, dobija se Android aplikacija koja je prikazana sledećom slikom.

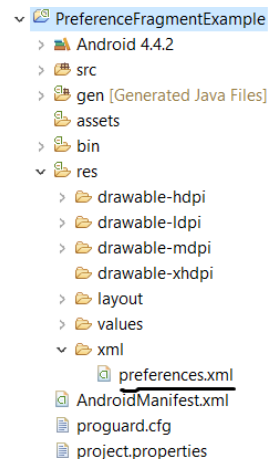


Slika-9 Fragment za dijalog

# PREFERENCEFRAGMENT KLASA

*U Android operativnom sistemu moguće je koristiti PreferenceActivity baznu klasu za prikazivanje aktivnosti koja korisniku omogućava opcije za prilagođavanje.*

Izbor opcija za personalizovanje aplikacija, vlastitim potrebama i navikama, omogućen je u Android operativnom sistemu. Bazna klasa, kojom su omogućene navedene funkcionalnosti, naziva se *PreferenceActivity* klasa. Novije verzije Android operativnog sistema donose još jednu klasu kojom je obezbeđeno da kreirana aplikacija ima navedene funkcionalnosti. Ova klasa je poznata pod nazivom *PreferenceFragment* klasa. Za demonstraciju, koristićemo primer u kojem će main.xml klasa ostati ne promenjena, ali će zato, u folderu koji će biti nazvan xml, biti kreirana datoteka *preferences.xml* (sledeća slika).



Slika-10 xml folder

Datoteka *preferences.xml* će biti izgrađena od sledećeg koda.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
  <PreferenceCategory android:title="Kategorija 1">
    <CheckBoxPreference
      android:title="Checkbox"
      android:defaultValue="false"
      android:summary="True ili False"
      android:key="checkboxPref" />
  </PreferenceCategory>
  <PreferenceCategory android:title="Kategorija 2">
    <EditTextPreference
      android:name="EditText"
      android:summary="Unesite string"
      android:defaultValue="[Unesite string ovde]"
      android:title="Edit Text"
      android:key="editTextPref" />
    <RingtonePreference
      android:name="Ringtone Preference"
      android:summary="Izaberite melodiju zvona"
      android:title="Melodija zvona"
      android:key="ringtonePref" />
  </PreferenceCategory>
  <PreferenceScreen
    android:title="Drugi ekran preferencija"
    android:summary="
      Kliknite da idete na sledeći ekran"
    android:key="secondPrefScreenPref">
    <EditTextPreference
      android:name="EditText"
      android:summary="Unesite string"
      android:title="Unesite string (drugi ekran)"
      android:key="secondEditTextPref" />
  </PreferenceScreen>
</PreferenceCategory>
</PreferenceScreen>
```

Slika-11 preferences.xml

# PREFERENCEFRAGMENT KLASA – JAVA KLASA PRIMERA

## *Fragment klasa nasleđuje baznu klasu PreferenceFragment*

Osnovna klasa aplikacije, klasa aktivnosti, data je sledećim kodom.

```
package net.learn2develop.PreferenceFragmentExample;

import android.app.Activity;

public class PreferenceFragmentExampleActivity extends Activity {
    /** Poziva se kada se aktivnost kreira. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        FragmentManager fragmentManager = getFragmentManager();
        FragmentTransaction fragmentTransaction =
            fragmentManager.beginTransaction();
        Fragment1 fragment1 = new Fragment1();
        fragmentTransaction.replace(android.R.id.content, fragment1);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
    }
}
```

Slika-12 Klasa aktivnosti primera

Fragment klasa mora da nasleđuje baznu klasu *PreferenceFragment* da bi funkcije prilagođavanja aplikacije korisničkim potrebama bile omogućene. Kod je prikazan sledećom slikom.

```
package net.learn2develop.PreferenceFragmentExample;

import android.os.Bundle;

public class Fragment1 extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //---Učitava preference iz XML datoteke---
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

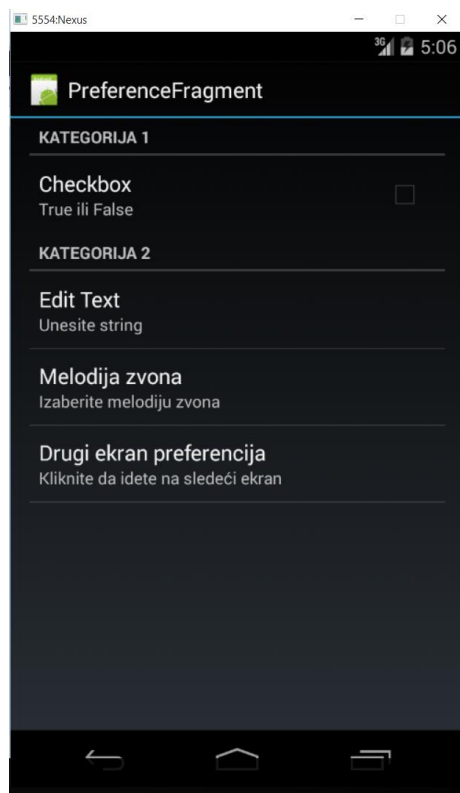
Slika-13 Fragment klasa



# PREFERENCEFRAGMENT KLASA – DEMONSTRACIJA PRIMERA

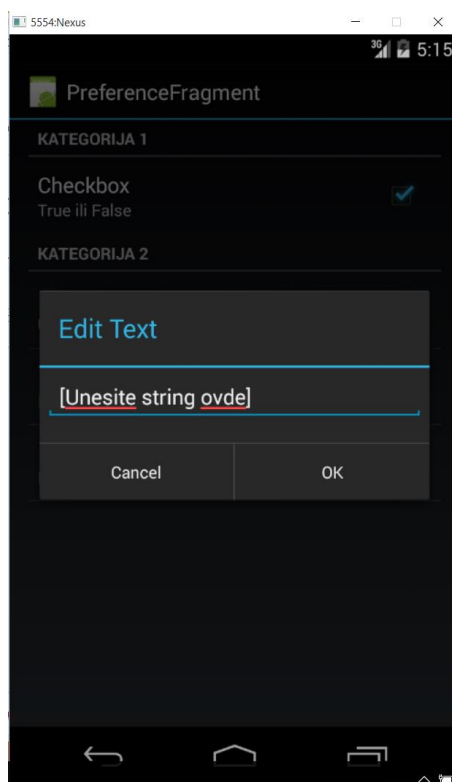
*Na ekranu se nalaze opcije koje omogućavaju korisniku da izabere željena podešavanja.*

Klikom na F11, program se prevodi i pokreće emulatorom (sledeća slika).



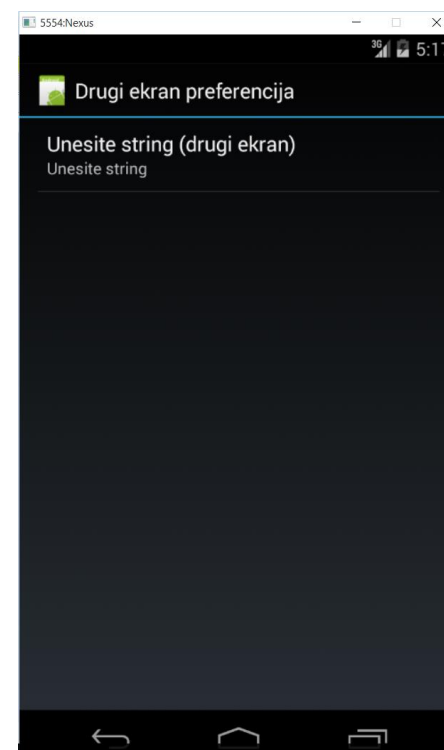
Slika-14 Početni ekran

Ako se izabere *Edit text* opcija, biće prikazane dodatne opcije (sledeća slika).



Slika-15 Edit text opcije

Ako se izabere opcija *Drugi ekran preferencija*, biće prikazan drugi ekran sa svojim opcijama.

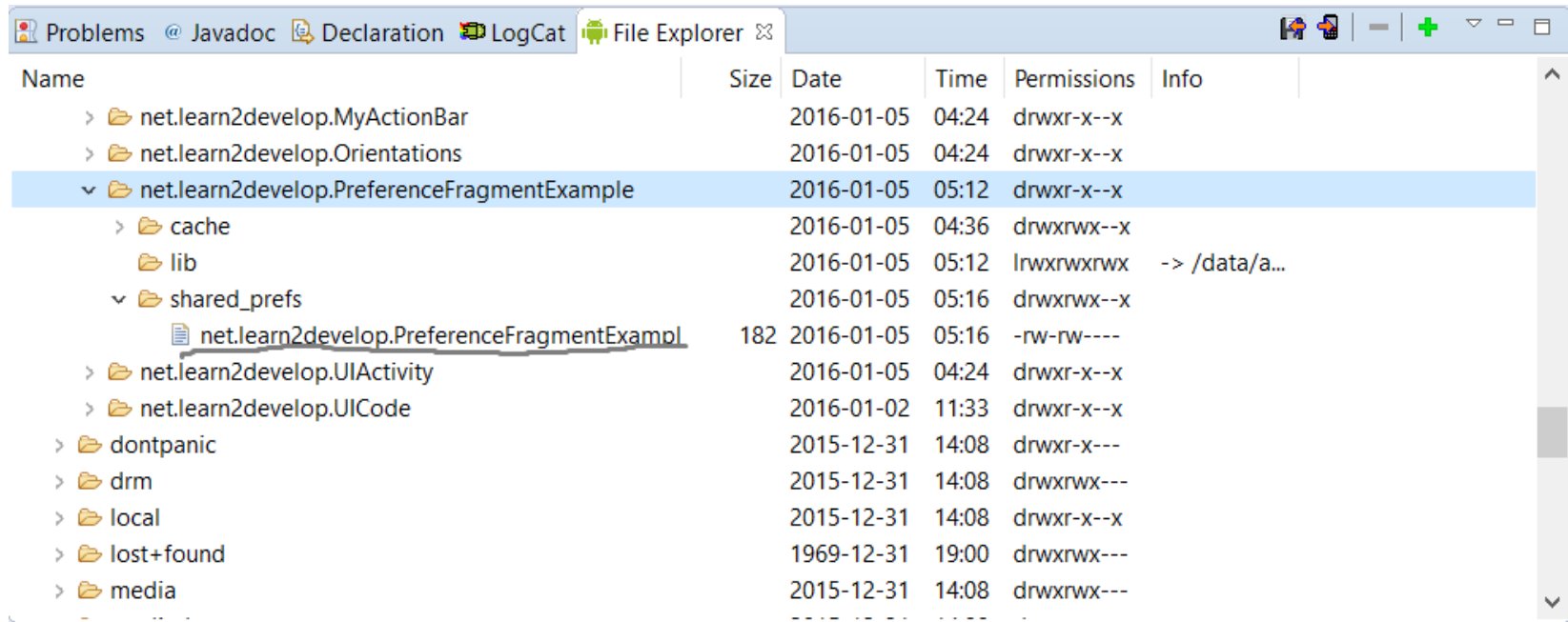


Slika-16 Drugi ekran preferencija

# ČUVANJE I LOCIRANE PREFERENCIJA

*Sva korisnička podešavanja biće sačuvana u posebnom dokumentu.*

Sva korisnička podešavanja biće sačuvana u posebnom dokumentu. Ukoliko se iskoristi opcija *File Explorer* (dostupna je u Dalvik Debug Monitor Service - DDMS prikazu) moguće je locirati datoteku sa snimljenim preferencijama. Potrebno je pratiti sledeću putanju: `data/data`, a zatim u tom folderu locirati naziv paketa (korišćeno je `com.metropolitan` ili `com.learn2develop`) sa nazivom ciljanog projekta. Ulaskom u ovaj podfolder, locira se folder `shered_prefs` u kojem je spakovana datoteka sa sačuvanim preferencijama (sledeća slika).



Name	Size	Date	Time	Permissions	Info
> net.learn2develop.MyActionBar		2016-01-05	04:24	drwxr-x--x	
> net.learn2develop.Orientations		2016-01-05	04:24	drwxr-x--x	
▼ net.learn2develop.PreferenceFragmentExample		2016-01-05	05:12	drwxr-x--x	
> cache		2016-01-05	04:36	drwxrwx--x	
lib		2016-01-05	05:12	lrwxrwxrwx	-> /data/a...
▼ shared_prefs		2016-01-05	05:16	drwxrwx--x	
net.learn2develop.PreferenceFragmentExampl	182	2016-01-05	05:16	-rw-rw----	
> net.learn2develop.UIActivity		2016-01-05	04:24	drwxr-x--x	
> net.learn2develop.UICode		2016-01-02	11:33	drwxr-x--x	
> dontpanic		2015-12-31	14:08	drwxr-x---	
> drm		2015-12-31	14:08	drwxrwx---	
> local		2015-12-31	14:08	drwxr-x--x	
> lost+found		1969-12-31	19:00	drwxrwx---	
> media		2015-12-31	14:08	drwxrwx---	

Slika-17 Datoteka preferencija

# PREFERENCEFRAGMENT KLASA - FUNKCIONISANJE

*Za učitavanje datoteke sa preferencijama, u fragment, koristi se specijalna metoda.*

Da bi preferencije bilo moguće kreirati, u nekoj Android aplikaciji, neophodno je prvo kreirati XML datoteku preferencija u kojoj se definišu različite perzistentne stavke aplikacije.

Da bi fragment preferencija bio kreiran, neophodno je kreirati klasu koja nasleđuje baznu klasu *PreferenceFragment*.

Učitavanje datoteke sa preferencijama u fragment omogućeno je metodom *addPreferencesFromSource()* koja je ugrađena u metodu *onCreate()* fragment klase aplikacije (sledeća slika).

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //---Učitava preference iz XML datoteke---
    addPreferencesFromResource(R.xml.preferences);
}
```

Slika-18 Učitavanje datoteke sa preferencijama

Na kraju, neophodno je pakovanje fragmenta preferencija na pozadinski stek pomoću metode *addToBackStack()* u klasi aktivnosti (sledeća slika). Na ovaj način, omogućeno je uklanjanje fragmenta klikom na taster *Back*.

```
FragmentManager fragmentManager = getFragmentManager();
FragmentManager fragmentManager =
    fragmentManager.beginTransaction();
FragmentManager fragmentManager = new fragmentManager();
FragmentManager fragmentManager.replace(android.R.id.content, fragmentManager);
FragmentManager fragmentManager.addToBackStack(null); ✓
FragmentManager fragmentManager.commit();
```

Slika-19 Dodavanje fragmenta na pozadinski stek