Mobilne komunikacije i datoteke

Mobilne aplikacije

Stevan Gostojić

Fakultet tehničkih nauka, Novi Sad

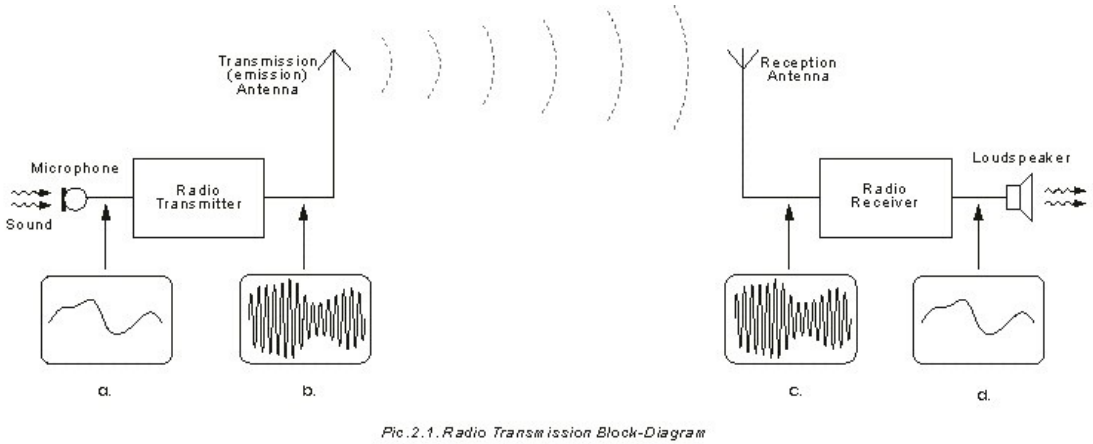
26. novembar 2024.

Agenda

Radio talasi

- Radio talasi su oscilacije elektromagnetnog polja u vremenu i prostoru.
- Slabljenje radio talasa zavisi od njegove frekvencije i karakteristika medija kroz koji se prostire.

Radio uređaji



Pic.2.1.Radio Transmission Block-Diagram

Figure 1: Prijemnik i predajnik.

- Pošiljalac
- Predajnik
- Antena
- Medij
- Antena
- Prijemnik
- Primalac

Analogni signal

Figure 2: Analogni signal.

- Analogni signal je kontinualni signal koji informacije prenosi kao promena amplitude, frekvencije i faze.

Digitalni signal

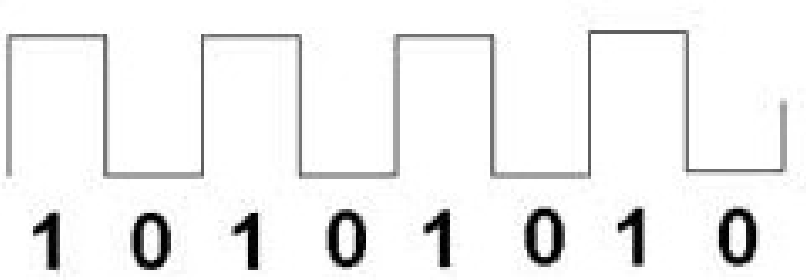


Figure 3: Digitalni signal.

- Digitalni signal je diskretan signal koji informacije prenosi kao niz znakova (logičkih nula i jedinica).

Modulacija

Modulacija je proces kojim se signal prilagođava karakteristikama prenosnog medija (transponuje se u područje frekvencija pogodnih za prenos radio talasima).

- Analogni postupci koriste se za molulisanje analognog signala
  - Amplitudna modulacija
  - Frekventna modulacija
  - Fazna modulacija
- Digitalni postupci koriste se za modulisanje digitalnog signala
  - Pulsna kodna modulacija
  - Delta modulacija

Agenda

Ćelijski sistem

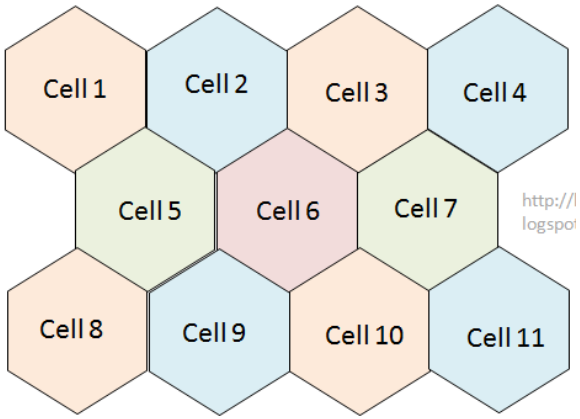


Figure 4: Ćelijski sistem.

- Određena teritorija je podeljena na ćelije.
- Svakoj ćeliji je dodeljen skup frekvencija koje su izabrane tako da minimizuju interferenciju sa susednim ćelijama.
- Skup frekvencija može se koristiti i u drugim ćelijama, sve dok te ćelije nisu susedne.
- Kada mobilni telefon pređe iz jedne ćelije u drugu ćeliju dok je poziv u toku, mreža će izdati naredbu mobilnom telefonu da promeni kanal (frekvenciju) i u isto vreme preusmeriti poziv na novi kanal.

Usluge GSM-a

- Telefonija (prenos govora)
- Short Message Service (SMS)
- Multimedia Messaging Service (MMS)
- Prenos podataka

SIM

- Subscriber Identity Module (SIM) je pametna kartica koja omogućava mobilnom telefonu da promeni pretplatnika i pretplatniku da promeni mobilni telefon.
- Implementira Java Card specifikaciju da bi omogućio interoperabilnost aplikacija.
- Pruža sigurno skladištenje:
  - Integrated Circuit Card Identifier (ICCID) - identifikator kartice
  - International Mobile Subscriber Identity (IMSI) - identifikator pretplatnika
  - i kriptografskog ključa (Ki) koji se koristi za autentifikaciju pretplatnika.

SIM

- Takođe skladišti:
  - Personal Identification Number (PIN) - lozinku za uobičajenu upotrebu
  - Personal Unblocking Code (PUK) - lozinku za otključavanje PIN-a
  - Service Provider Name (SPN)
  - Local Area Identity (LAI)
  - broj SMSC-a
  - broj za hitne slučajeve
  - spisak uluga kojima pretplatnik može da pristupi
  - itd.
- Nudi i dodatne funkcije kao što je skladištenje telefonskog imenika i tekstualnih poruka.

Agenda

Telefonija

- Osnovna usluga mobilne mreže je mobilna telefonija (prenos govora).
- Međutim, iz bezbednosnih razloga nije moguće napraviti "in call" aktivnost.
- Android API omogućava:
  - korišćenje podrazumevane "in call" aktivnosti za obavljanje telefonskih poziva
  - pristup podacima o telefonu (tip, identifikator, verzija softvera, telefonski broj)
  - pristup podacima o SIM kartici (stanje, država i ime operatora, serijski broj)
  - pristup podacima o mreži (država, identifikator operatora, ime mreže, tip mreže)
  - pristup podacima o prenosu podataka (stanje i trenutna aktivnost)
  - reagovanje na promenu stanja mreže, poziva, lokacija ćelije, snage signala, aktivnosti i stanja prenosa podataka, itd.

AndroidManifest.java

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application ... >
4     <uses-permission android:name="android.permission.CALL_PHONE"/>
5   </application>
6 </manifest>
7
```

ExampleActivity.java

```
1  public void dialNumber() {
2    Intent intent =
3              new Intent(Intent.ACTION_DIAL, Uri.parse("tel:1234567"));
4    startActivity(intent);
5  }
6
7  // ACTION_DIAL opens the dialer app with no permissions required
8  // ACTION_CALL initiates phone call and requires permission requests
       statically (in manifest file) and dynamically (at run-time)
       because protection level for this permission is 'dangerous'
9
10
```

Agenda

SMS

- Short Message Service (SMS) je tehnologija koja omogućava slanje i primanje kratkih poruka između mobilnih telefona.
- Podržavaju je (skoro) svi mobilni telefoni.
- Jedna SMS poruka može da sadrži najviše 140 bajtova (160 znakova ukoliko se koristi 7-bitno kodiranje, 70 znakova ukoliko se koristi 16-bitno kodiranje).
- Pored teksta, SMS poruke mogu da prenose i binarne podatke.

PDU mode SMS

- Konkatenirane SMS poruke ili PDU (protocol data unit) mode SMS poruke mogu da sadrže više od 140 bajtova.
- Mobilni telefon pošiljaoca deli dugačku poruku u manje delove i šalje svaki deo kao pojedinačnu SMS poruku.
- Mobilni telefon primaoca spaja pojedinačne SMS poruke u dugačku poruku.

SMSC

- Short Message Service Center (SMSC) upravlja SMS operacijama celularne mreže (njegova glavna funkcija je rutiranje SMS poruka).
- Kada mobilni telefon pošalje SMS poruku, ona stiže u SMS centar. SMS centar prosleđuje SMS poruku primaocu.
- Ako je primalac nedostupan (mobilni telefon je isključen ili nema domet), SMS centar skladišti poruku i prosleđuje je primaocu kada postane dostupan.

SMS Gateway

- Pre nego što stigne do odredišta, SMS poruka može da prođe kroz SMS gateway i druge SMS centre.
- SMS Gateway omogućava različitim operaterima mobilne telefonije da povežu SMS centre i razmenjuju SMS poruke.

MMS

- Multimedia Messaging Service (MMS) je proširenje SMS-a.
- MMS omogućava formatiranje teksta i slanje i primanje multimedijalnih poruka (poruka koje sadrže fotografije, audio i video).

SMS i Android

Postoje dva načina slanja SMS poruka:

- putem podrazumevane SMS aplikacije ili
- putem SMSManager servisa.

ExampleActivity.java (slanje kroz podrazumevanu aplikaciju)

```
1 public void sendSMS() {
2   Intent smsIntent = new Intent(Intent.ACTION_SENDTO, Uri.parse("sms:+381641234567"));
3   smsIntent.putExtra("sms_body", "Please call me as soon as possible");
4   startActivity(smsIntent);
5 }
6
```

ExampleActivity.java (slanje kroz podrazumevanu aplikaciju)

```
1  public void sendMMS() {
2    // Get the URI of a piece of media to attach.
3    Uri attachedUri = Uri.parse("content://media/external/images/media/1");
4    // Create a new MMS intent
5    Intent mmsIntent = new Intent(Intent.ACTION_SEND);
6    mmsIntent.putExtra("sms_body", "Please see the attached image");
7    mmsIntent.putExtra("address", "+381641234567");
8    mmsIntent.putExtra(Intent.EXTRA_STREAM, attachedUri);
9    mmsIntent.setType("image/png");
10   startActivity(mmsIntent);
11 }
12
```

AndroidManifest.xml (slanje putem SmsManager servisa)

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application ... >
4     <uses-permission android:name="android.permission.SEND_SMS"/>
5   </application>
6 </manifest>
7
8 <!-- This permission should be requested in run-time as well because
      its level is 'dangerous' -->
9
```

ExampleActivity.java (slanje putem SmsManager servisa)

- Do API level 31:

```
1 public sendSMS() {
2   SmsManager smsManager = SmsManager.getDefault();
3   String to = "+381641234567";
4   String message = "Android supports programmatic SMS messaging!";
5   smsManager.sendTextMessage(to, null, message, null, null);
6 }
```

- Od API level 31:

```
1 public sendSMS() {
2   SmsManager smsManager = getSystemService(SmsManager.class);
3   String to = "+381641234567";
4   String message = "Android supports programmatic SMS messaging!";
5   smsManager.sendTextMessage(to, null, message, null, null);
6 }
```

SMS

Parameters	Meaning
destinationAddress	The address to send the message to
serviceCenterAddress	The service center address (or null to use the default SMSC).
text	The body of the message to send.
sentIntent	If not null this PendingIntent is broadcast when the message is successfully sent, or failed.
deliveryIntent	If not null this PendingIntent is broadcast when the message is delivered to the recipient.

Table 1: Parametri `sendTextMessage` metode.

MMS/Data/Multipart Messages

- `sendDataMessage` (sends a data based SMS to a specific application port)
- `sendMultimediaMessage` (sends an MMS message)
- `sendMultipartTextMessage` (sends a multi-part text based SMS)

Agenda

Networking

- Umrežavanje mobilnih uređaja je slično kao i kod računara i računarske opreme.
- Bez obzira na tip konekcije za razmenu podataka, na raspolaganju su nam protokoli sa viših OSI nivoa.
- Nekim lokacijama nije potrebno pristupati često (slika, audio zapis, video klip i sl.) dok se neke lokacije mogu intenzivno koristiti (npr. mail server, web servisi).

Primer preuzimanja podataka sa neke web adrese

```
1     InputStream is = null;
2     try {
3       URL url = new URL("https://some.address.com/some/path");
4       URLConnection conn = url.openConnection();
5       is = conn.getInputStream();
6       ByteArrayOutputStream os = new ByteArrayOutputStream();
7       byte[] buff = new byte[1024];
8       int read;
9       while ((read = is.read(buff)) > 0) {
10        os.write(buff,0,read);
11      }
12      return os.toString();
13    } catch (Exception e) {
14      e.printStackTrace();
15    } finally {
16      is.close();
17    }
18
```

Web servisi

- Za udaljeno izvršavanje operacija nad podacima se obično koriste REST servisi.
- Podaci se najčešće prenose u JSON formatu što olakšava njihovu serijalizaciju/deserijalizaciju.
- Iako je ovaj način komunikacije relativno jednostavan za implementiranje, u praksi se koriste gotove biblioteke koje dodatno olakšavaju korišćenje REST servisa.

Retrofit

- Retrofit je HTTP klijent za Android i Javu.
- Omogućava rad sa sinhronim i asinhronim pozivima, pri čemu je na Android platformi asinhrona varijanta praktičnija jer se na taj način ne blokira glavna UI nit.
- Retrofit biblioteka ne vrši serijalizaciju i deserijalizaciju JSON objekata, pa je u te svrhe potrebno uključiti i druge biblioteke u projekat.

build.gradle

```
dependencies {
    implementation 'com.squareup.retrofit2:retrofit:2.11.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.11.0'
    ...
}

/*
  converter-gson enables retrofit to use the gson library.
  converter-gson depends on gson library that performs
                 (de)serialization of JSON objects.
*/

```

Model podataka

- Za konverziju između JSON i Java objekata potrebno je u projektu definisati model podataka.
- Ovaj model predstavljaju POJO klase sa odgovarajućim anotacijama koje povezuju atribute ovih klasa sa odgovarajućim poljima u JSON objektima.
- Na adresi `https://www.jsonschema2pojo.org/` se nalazi online alat za automatsko generisanje anotiranih klasa na osnovu uzorka JSON podataka. Potrebno je izabrati:
  - Source Type: JSON
  - Annotation Style: Gson

Book.java

```java
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;

public class Book {

    @SerializedName("id")
    @Expose
    private Integer id;

    @SerializedName("title")
    @Expose
    private String title;

    @SerializedName("author")
    @Expose
    private String author;

}

```

Endpoint

- Nakon što smo povezali Java model sa JSON modelom potrebno je definisati endpoint pomoću metoda u Java interfejsu.
- To se takođe postiže pomoću anotacija uz metode i njihove parametre tako što označavaju na koji način učestvuju u HTTP request-u i response-u.
- Pri tome se putanje do endpointa navode u relativnom zapisu dok se osnovni deo URL-a setuje pri inicijalizaciji Retrofit instance.

BooksEndpoint.java

```
1  import java.util.List;
2  import retrofit2.Call;
3  import retrofit2.http.Body;
4  import retrofit2.http.GET;
5  import retrofit2.http.POST;
6  import retrofit2.http.Path;
7
8  public interface BooksEndpoint {
9
10     @GET("books")
11     Call<List<Book>> getBooks();
12
13     @GET("books/{id}")
14     Call<Book> getBook(@Path("id") int id);
15
16     @POST("books")
17     Call<Book> createBook(@Body Book book);
18
19 }
20 // return type of methods is Call<T>
21 // where T is a type of an object the method should return.
22
```

Anotacije

Annotation	Description
@GET @POST @PUT @DELETE	Type of HTTP method
@Path	Named replacement in a URL path segment
@Query	Query parameter appended to the URL
@Body	Controls the request body
@Header	Specifies the header parameter
@Headers	Predefines the header parameters

Table 2: Objašnjenja anotacija endpoint-a.

Instanciranje Retrofit objekta

- Da bi smo koristili Retrofit potrebno je da najpre instanciramo objekat klase `Retrofit` putem `Builder`-a.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(BASE_URL)
    .addConverterFactory(GsonConverterFactory.create())
    .build();
```

- `Builder`-u smo setovali:
  - osnovni URL (na koji će se dodavati relativne putanje definisane u interfejsu endpoint-a) i poželjno je definisati ga kao konstantu

```
public static final String BASE_URL = "http://.../";
```

  - konverter koji će se koristiti za (de)serijalizaciju JSON objekata (u ovom slučaju to je konverter iz Gson biblioteke).

Kreiranje poziva

- Na osnovu interfejsa koji smo definisali, Retrofit omogućava kreiranje objekta koji reprezentuje servis i omogućava pozivanje metoda koje smo deklarisali.

```
BooksEndpoint be = retrofit.create(BooksEndpoint.class);
```

- Nad ovim objektom kreiramo poziv neke od metoda na sledeći način:

```
Call<List<Book>> call = be.getBooks();
```

- Objekat tipa `Call<T>` reprezentuje poziv ka servisu i dozvoljava sinhrono i asinhrono izvršavanje.
- Za sinhrono izvršavanje se koristi metoda `execute()`.
- Za asinhrono izvršavanje se koristi metoda `enqueue()`.

Sinhrono izvršavanje poziva

```
1 Response<List<Book>> response = call.execute();
2 List<Book> books = response.body();
3
```

Asinhrono izvršavanje poziva

```
call.enqueue(new Callback<List<Book>>() {
    @Override
    public void onResponse(Call<List<Book>> call, Response<List<Book>> response) {
        // success
        List<Book> books = response.body();
    }
    @Override
    public void onFailure(Call<List<Book>> call, Throwable t) {
        // failure
    }
});

```

Inicijalizacija Retrofit biblioteke i korišćenje asinhronog poziva

```
1  // base url is defined as a global constant
2  public static final String BASE_URL = "http://some.address.com/";
3
4    // retrofit instance is created using the Builder
5    // base url and (de)serializer are initially set
6    Retrofit retrofit = new Retrofit.Builder()
7      .baseUrl(BASE_URL)
8      .addConverterFactory(GsonConverterFactory.create())
9      .build();
10
11   // the object representing service is created using retrofit and the interface
12   BooksEndpoint be = retrofit.create(BooksEndpoint.class);
13   // request is created and stored in object representing this call
14   Call<List<Book>> call = be.getBooks();
15
16   // the request is sent and response is processed asynchronously
17   call.enqueue(new Callback<List<Book>>() {
18     @Override
19     public void onResponse(Call<List<Book>> call, Response<List<Book>> response) {
20       int status = response.code();
21       List<Book> books = response.body();
22       // ...
23     }
24     @Override
25     public void onFailure(Call<List<Book>> call, Throwable t) {
26       // the request is failed
27     }
28   });
29
```

Asinhrono učitavanje slika

- Kada je potrebno da se neka grafička datoteka sa veba učita u pozadini korisno je upotrebiti Picasso biblioteku.
- Najpre je potrebno uključiti odgovarajuću zavisnost:

```
implementation 'com.squareup.picasso:picasso:2.5.2'
```

- Zatim se učitavanje slike u neki ImageView pogled može postići na sledeći način:

```
ImageView iv = findViewById(R.id.imageView);
Picasso.with(context).load("http://...").into(iv);
```

Pregled sadržaja

Deljena podešavanja

- Deljena podešavanja (SharedPreferences) olakšavaju perzistentno skladištenje prostih tipova podataka
- Ti podaci se skladište u datoteci kao uređeni parovi (ključ, vrednost)

Deljena podešavanja

- Deljenim podešavanjima se može pristupiti metodom `SharedPreferences getSharedPreferences(String name, int mode)`
- Ova metoda je definisana u klasi `Context`, pa je samim tim dostupna i u okviru njenih naslednica, kao što su `Activity` i `Service`.
- Moguće je koristiti više skupova deljenih podešavanja čija imena se navode kao parametar (`name`).

Deljena podešavanja

Konstanta	Opis
`MODE_PRIVATE`	The created file can only be accessed by the calling application

Table 3: Vrednosti parametra "mode".

Ostali režimi su označeni kao `deprecated`.

Deljena podešavanja

- U klasi `Activity` je definisana metoda
  `SharedPreferences getPreferences(int mode)`
  koja omogućava pristup skupu podešavanja te aktivnosti.
- Za razliku od metode `getSharedPreferences`, ova metoda koristi podrazumevan naziv skupa podešavanja koji čine naziv paketa i naziv klase u kojoj je aktivnost implementirana.

Deljena podešavanja

- Uz korišćenje paketa `androidx.preferences` na raspolaganju je i pristup podrazumevanom skupu podešavanja:

```
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
```

pri čemu je naziv tog skupa sastavljen od naziva paketa i sufiksa "_preferences".

Metoda	Naziv datoteke	Opis
`getSharedPreferences("some_name",MODE_PRIVATE)`	some_name.xml	Poizvoljan naziv skupa podešavanja
`getPreferences(MODE_PRIVATE)`	activityname.xml	Podešavanja koja se odnose na jednu aktivnost
`PreferenceManager.getDefaultSharedPreferences(this)`	packagename_preferences.xml	Podešavanja koja se odnose na kontekst (podrazumevano na paket)

Table 4: Načini pristupa podešavanjima.

Deljena podešavanja - upis vrednosti

Vrednosti prostog tipa T mogu se zapisati u tri koraka:

1. Pozvati `edit()` metodu koja započinje transakciju
2. Dodati vrednost(i) tipa T metodama oblika `SharedPreference.Editor putT(String key, T value)`
3. Pozvati `commit()` metodu koja završava transakciju u sinhronom modu (i vraća rezultat o uspehu zapisivanja podataka) ili `apply()` koja to radi u asinhronom modu.

U zavisnosti od tipa T, na raspolaganju su metode:
`putBoolean`, `putFloat`, `putInt`, `putLong`, `putString`, `putStringSet`.
Ove metode vraćaju tip `SharedPreference.Editor` tako da je olakšano ulančavanje višestrukih upisa vrednosti.

Deljena podešavanja

- Zapisane vrednosti mogu se pročitati metodama oblika `T getT(String key, T defaultValue)`
- U zavisnosti od tipa T, na raspolaganju su metode: `getBoolean`, `getFloat`, `getInt`, `getLong`, `getString`, `getStringSet`
- S obzirom na to da su nazivi skupova podešavanja i nazivi pojedinačnih vrednosti predstavljeni stringovima, može biti korisno njihovo eksternalizovanje (`R.string.xyz`) jer se u projektu mogu koristiti po više puta.

ExampleActivity.java

```
1 SharedPreferences settings = getPreferences(Context.MODE_PRIVATE);
2 SharedPreferences.Editor editor = settings.edit();
3 editor.putBoolean("silentMode", silentMode);
4 editor.putInt("refreshInterval", refreshInterval)
5 editor.commit();
6
```

Ili kraće (sa ulančanim pozivima metoda):

```
1 getPreferences(MODE_PRIVATE).edit()
2         .putBoolean("silentMode", silentMode)
3         .putInt("refreshInterval", refreshInterval)
4         // ...
5         .commit();
6
```

ExampleActivity.java

```
1 SharedPreferences settings = getPreferences(Context.MODE_PRIVATE);
2 boolean silentMode = settings.getBoolean("silentMode", false);
3 int refreshInterval = settings.getInt("refreshInterval", 1000);
4
```

PreferenceActivity

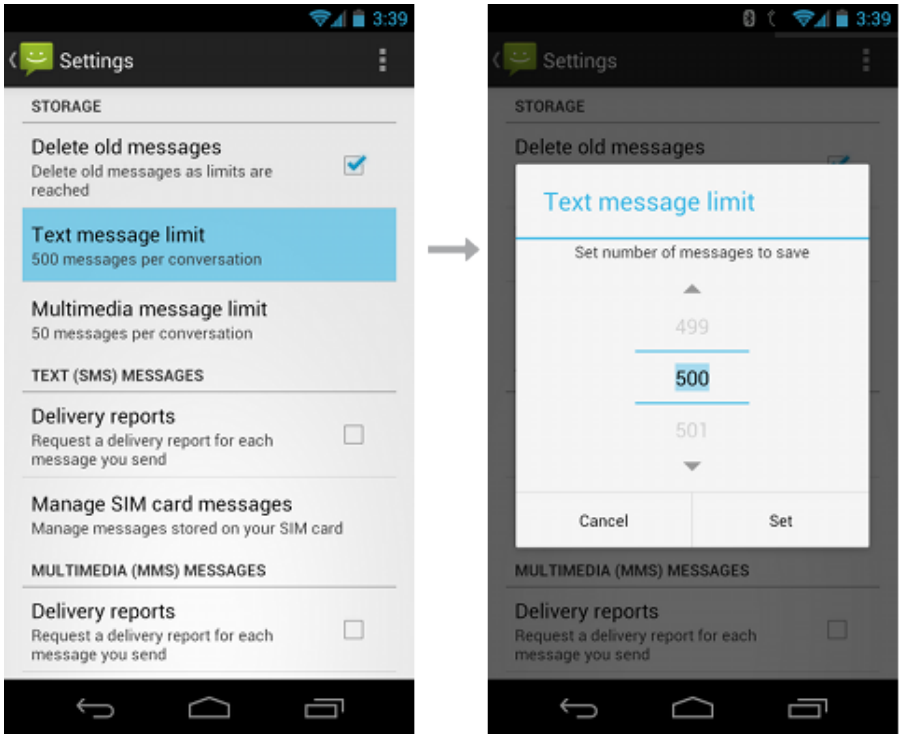


Figure 5: PreferenceFragmentCompat.

- Mnoge aplikacije omogućavaju korisnicima konfigurisanje
- U tu svrhu treba koristiti `PreferenceFragmentCompat` kako bi korisnici imali konzistentan grafički korisnički interfejs (i da bi sebi olakšali posao)

Deljena podešavanja

- Za upravljanje podešavanjima putem korisničkog interfejsa je namenjena klasa `PreferenceFragmentCompat` čijim nasleđivanjem definišemo sopstvene fragmente za rad sa podešavanjima.
- `PreferenceFragmentCompat` je definisan u paketu koji je potrebno dodati u Gradle skriptu projekta kao zavisnost:
  `implementation 'androidx.preference:preference:1.2.0'`
- Ovakav fragment dinamički kreira svoj izgled ako pri njegovom kreiranju pozovemo metodu `setPreferencesFromResource` kojom se učitava XML resurs sa definisanim podešavanjima (obrađeno na predavanjima GUI III).
- Postupak kreiranja pomenute aktivnosti i fragmenta Android Studio olakšava pomoću wizard-a: File -> New -> Activity -> Settings Activity.

AndroidManifest.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest ... >
3   <application ... >
4     <activity android:name=".SettingsActivity">
5       <!-- ... -->
6     </activity>
7   </application>
8 </manifest>
9
```

SettingsActivity.java

```
1  public class SettingsActivity extends AppCompatActivity {
2
3      @Override
4      protected void onCreate(Bundle savedInstanceState) {
5          super.onCreate(savedInstanceState);
6          setContentView(R.layout.settings_activity);
7          if (savedInstanceState == null) {
8              getSupportFragmentManager()
9                      .beginTransaction()
10                     .replace(R.id.settings, new SettingsFragment())
11                     .commit();
12         }
13         // display back arrow in title bar
14         ActionBar actionBar = getSupportActionBar();
15         if (actionBar != null) {
16             actionBar.setDisplayHomeAsUpEnabled(true);
17         }
18     }
19     // implementation of fragment that displays preferences as GUI
20     public static class SettingsFragment extends PreferenceFragmentCompat {
21         @Override
22         public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
23           // preferences name can by specified by:
24           // getPreferenceManager().setSharedPreferencesName("some_settings_name");
25           // or default name will be used (package name with "_preferences" suffix)
26
27             setPreferencesFromResource(R.xml.app_preferences, rootKey);
28         }
29     }
30 }
31
```

layout/settings_activity.xml

```
1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      android:layout_width="match_parent"
3      android:layout_height="match_parent">
4
5      <FrameLayout
6          android:id="@+id/settings"
7          android:layout_width="match_parent"
8          android:layout_height="match_parent" />
9
10 </LinearLayout>
11
```

xml/app_preferences.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen xmlns:app="http://schemas.android.com/apk/res-auto">
3
4    <CheckBoxPreference
5      app:key="pref_sync"
6      app:title="@string/pref_sync"
7      app:summary="@string/pref_sync_summ"
8      app:defaultValue="true" />
9
10   <ListPreference
11     app:dependency="pref_sync"
12     app:key="pref_syncConnectionType"
13     app:title="@string/pref_syncConnectionType"
14     app:entries="@array/pref_syncConnectionTypes_entries"
15     app:entryValues="@array/pref_syncConnectionTypes_values"
16     app:defaultValue="@string/pref_syncConnectionTypes_default" />
17
18 </PreferenceScreen>
19
```

Datoteka sa vrednostima podešavanja

- Podešavanja aplikacije se na uređaju čuvaju u folderu: data/data/package_name/shared_prefs

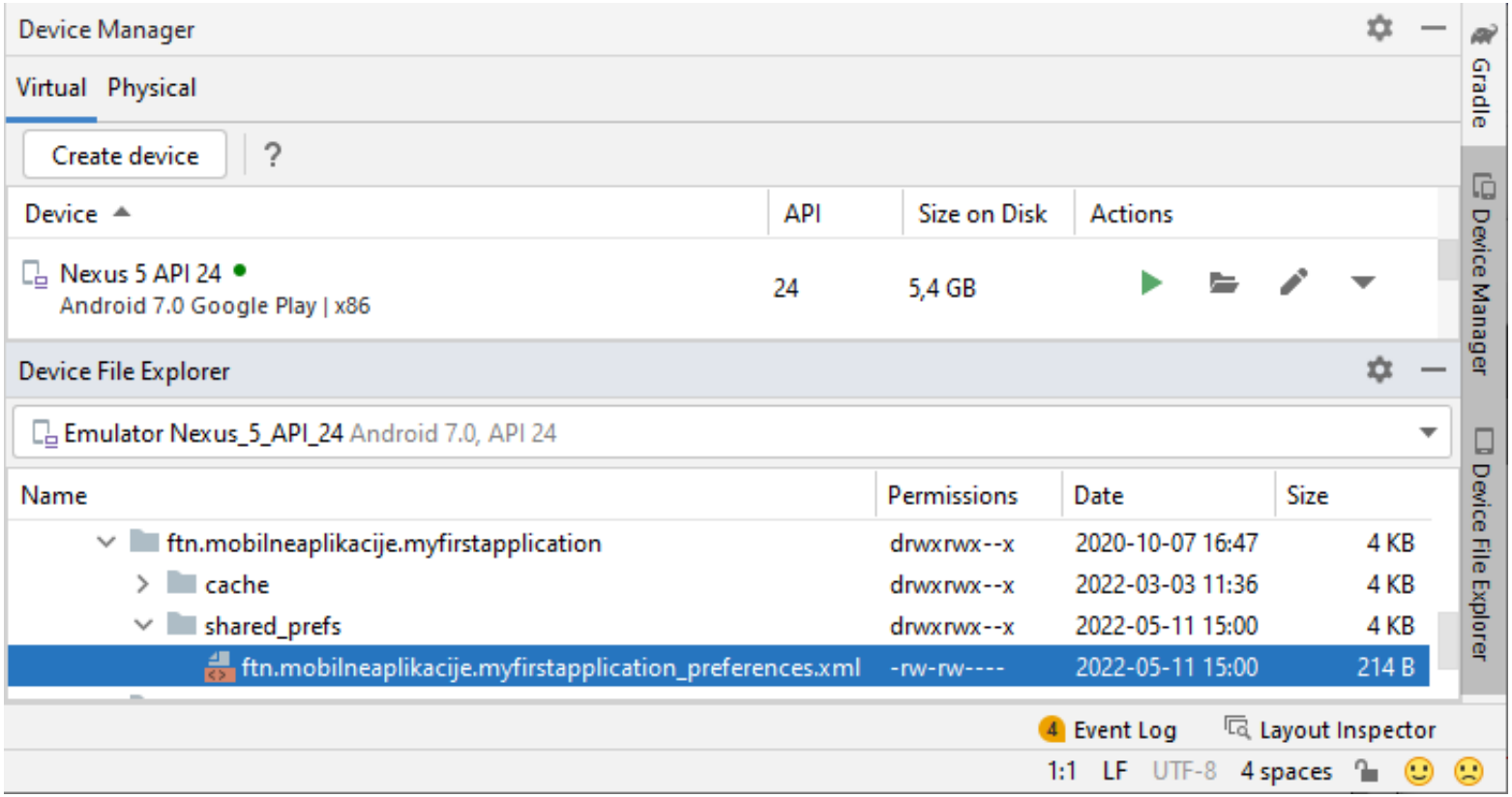


Figure 6: Datoteka sa podešavanjima

Pristup datotekama putem ADB

- Alat ADB omogućava pristup fajl sistemima uređaja (fizičkim i virtuelnim) koji su povezani na računar.

Figure 7: Izlistavanje sadržaja datoteke sa podešavanjima

Pregled sadržaja

Datoteke

- Podaci koji se nalaze u operativnoj memoriji se ne čuvaju kada se uništi proces
- Komponente koje se nalaze u različitim procesima ne mogu da razmenjuju podatke koji se nalaze u operativnoj memoriji (ne dele adresni prostor)
- Najjednostavniji način da se prevaziđu ova ograničenja je korišćenje datoteka

Datoteke

- Za rad sa datotekama koriste se klase iz java.io paketa (na isti način na koji se koriste u Java SE)
- Međutim, mogu se koristiti metode klase Context koje olakšavaju pristup internom i/ili eksternom skladištu podataka, rad sa privremenim datotekama i upravljanje pravima pristupa
  - `FileInputStream openFileInput(String name)`
  - `FileOutputStream openFileOutput(String name, int mode)`
  - `String[] fileList()` // vraća listu datoteka u folderu kojem samo tekuća aplikacija ima pristup (privatan folder)
  - `boolean deleteFile(String name)` // briše navedenu datoteku u privatnom folderu
  - `File getDir(String name, int mode)` // kreira ili pristupa folderu unutar privatnog foldera
  - `File getCacheDir()` //folder za privremene fajlove
  - `File getExternalCacheDir()` //folder za privremene fajlove na eksternom skladištu
  - `File getFilesDir()` //privatan folder - kome samo tekuća aplikacija ima pristup
  - `File getExternalFilesDir(String type)` // folder kome i druge aplikacije mogu pristupiti

ExampleService.java

```
1  private void write() {
2    FileOutputStream fos = null;
3    try {
4      fos = openFileOutput("test.txt", Context.MODE_PRIVATE);
5      fos.write(bytes);
6    } catch (FileNotFoundException e) {
7      Log.e(Constants.LOG_TAG, "File not found", e);
8    } catch (IOException e) {
9      Log.e(Constants.LOG_TAG, "IO problem", e);
10   } finally {
11     try {
12       fos.close();
13     } catch (IOException e) {
14     }
15   }
16 }
17
```

Datoteke

Konstanta	Opis
MODE_PRIVATE	The created file can only be accessed by the calling application
MODE_APPEND	if the file already exists then write data to the end of the existing file instead of erasing it
MODE_WORLD_READABLE	Allow all other applications to have read access to the created file (deprecated)
MODE_WORLD_WRITEABLE	Allow all other applications to have write access to the created file (deprecated)

Table 5: Vrednosti parametra "mode".

ExampleService.java

```
1  private void read() {
2    FileInputStream fis = null;
3    Scanner scanner = null;
4    StringBuilder sb = new StringBuilder();
5
6    try {
7      fis = openFileInput("test.txt");
8      scanner = new Scanner(fis);
9      while (scanner.hasNextLine()) {
10       sb.append(scanner.nextLine() + LINE_SEP);
11     }
12   } catch (FileNotFoundException e) {
13     Log.e(Constants.LOG_TAG, "File not found", e);
14   } finally {
15     if (fis != null) {
16       try {
17         fis.close();
18       } catch (IOException e) {
19       }
20     }
21     if (scanner != null) {
22       scanner.close();
23     }
24   }
25 }
26
```

Datoteke

- Interno skladište podataka se nalazi u mobilnom uređaju
  - Uvek je dostupno
  - Obično je manjeg kapaciteta (i nije ga moguće proširiti)
  - Privatno je
- Eksterno skladište podataka se (obično) nalazi na SD kartici
  - Nije uvek dostupno
  - Obično je većeg kapaciteta (i moguće ga je proširiti)
  - Javno je

ExampleService.java

```
1  String state = Environment.getExternalStorageState();
2  if (Environment.MEDIA_MOUNTED.equals(state)) {
3      // We can read and write the media
4  } else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
5      // We can only read the media
6  } else {
7      // Something else is wrong. It may be one of many other states,
8      // but all we need to know is we can neither read nor write
9  }
10
```

Datoteke

- Privremene datoteke treba skladištiti u cache direktorijumu (Android ih automatski briše kada ponestane slobodnog prostora)
  - `File getCacheDir()`
  - `File getExternalCacheDir()`
- Datoteke koje deli više aplikacija treba snimiti u javni eksterni direktorijum
  - `File getExternalFilesDir(String type)`
  - `File[] getExternalFilesDirs(String type)`

Datoteke

- Pri korišćenju eksternog skladišta potrebno je obezbediti statičke (putem manifest fajla) i dinamičke (putem dijaloga) permisije:
  - `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`
  - `<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />`
- Permisije nisu neophodne kada se koriste direktorijumi namenjeni tekućoj aplikaciji odn. direktorijumi koje vraćaju metode `getExternalFilesDir()` i `getExternalCacheDir()`.

Konstante klase Environment

Konstanta	Opis
DIRECTORY_ALARMS	Standard directory in which to place audio files for alarms
DIRECTORY_DCIM	The traditional location for pictures and videos when mounting the device as a camera.
DIRECTORY_DOCUMENTS	Standard directory in which to place documents that have been created by the user.
DIRECTORY_DOWNLOADS	Standard directory in which to place files that have been downloaded by the user.
DIRECTORY_MOVIES	Standard directory in which to place movies that are available to the user.

Table 6: Tip javnog eksternog direktorijuma.

Konstante klase Environment

Konstanta	Opis
DIRECTORY_MUSIC	Standard directory in which to place any audio files that should be in the regular list of music for the user.
DIRECTORY_NOTIFICATIONS	Standard directory in which to place any audio files that should be in the list of notifications that the user can select.
DIRECTORY_PICTURES	Standard directory in which to place pictures that are available to the user.
DIRECTORY_PODCASTS	Standard directory in which to place any audio files that should be in the list of podcasts that the user can select (not as regular music).
DIRECTORY_RINGTONES	Standard directory in which to place any audio files that should be in the list of ringtones that the user can select (not as regular music).

Table 7: Tip javnog eksternog direktorijuma.