MOBILNE APLIKACIJE

Vežbe 8

Deljena podešavanja i rad sa datotekama

2022/2023

Sadržaj

1. Deljena podešavanja.....3
2. Rad sa datotekama.....8
3. Snimanje fotografije i video zapisa.....9
5. Domaći.....12

1. Deljena podešavanja

Deljena podešavanja (*SharedPreferences*) olakšavaju perzistentno skladištenje prostih tipova podataka. Podatke čuvamo u datoteci kao uređene parove (ključ, vrednost).

Kreirali smo klasu *ReviewerPreferenceActivity* i u nju smo smestili fragment *PrefsFragment* (slika 1). Ovaj fragment će predstavljati naša podešavanja. Kreirali smo fragment jer se preporučuje da ekran sa podešavanjima bude baš fragment.

```
public static class PrefsFragment extends PreferenceFragmentCompat {

    private static PrefsFragment newInstance() {
        Bundle args = new Bundle();

        PrefsFragment fragment = new PrefsFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
        addPreferencesFromResource(R.xml.preferences);
    }
}
```

Slika 1. *PrefsFragment*

Ovaj fragment neće imati svoj layout, već ćemo koristiti posebnu specifikaciju, koja opisuje koja podešavanja imamo (slika 2). U elementu <PreferenceScreen> se definišu samo elementi koji su sastavni deo podešavanja (nema layout-a).

PreferenceCategory

Ovaj element služi za grupisanje drugih elemenata. Grupe su u podešavanjima malo odvojene, razdvojene linijom. Atributi koje navodimo su: naslov i ključ grupe.

CheckBoxPreference

Ovaj element može da prikazuje dva stanja (polje je označeno ili nije).

Atributi koje navodimo su:

- defaultValue – kada pokrenemo aplikaciju ovo će biti podrazumevana vrednost, u našem primeru vrednost je false
- key - ključ
- summary – kratak opis
- title – naziv

ListPreference

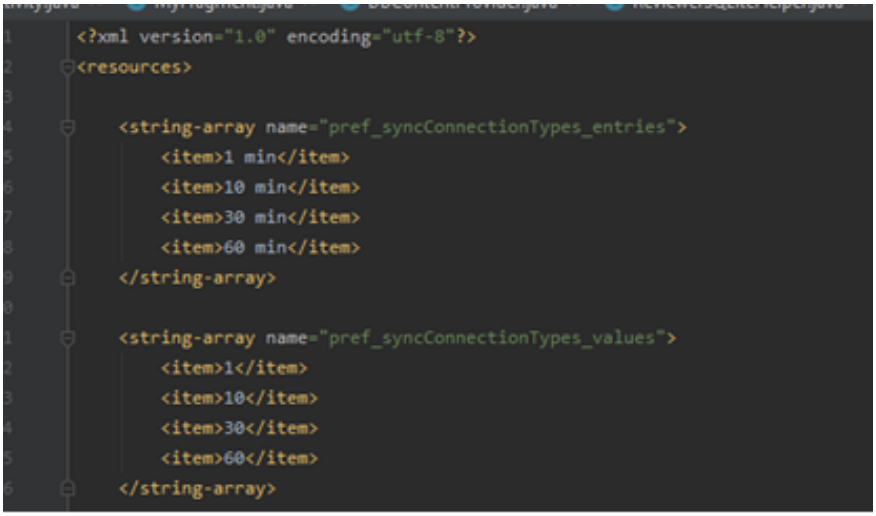
Pre elementa ListPreference uglavnom sledi checkBox element za koji vezujemo listu. Bitni atributi koje navodimo za listu su:

- dependency – na ovaj atribut postavljamo ključ checkBoxPreference elementa za koji želimo da vežemo ovu listu. Ako korisnik ne čekira prethodno definisan checkBoxPreference, onda će lista biti onemogućena za manipulaciju. Primer se vidi na slici 4. Na slici levo checkBox nije označen, a desno jeste.
- summary - navodimo kog tipa će biti elementi te liste koje prikazujemo (%s-string).
- entries - govori šta će korisniku biti prikazano kad se otvori lista.
- entryValues - su vrednosti koje će biti upisane u sharedPreferences za odgovarajući element.

Sledi da liste entries i entryValues moraju biti iste dužine (za svaki entry koji se prikazuje, postoji odgovarajući entryValue, koji se postavlja kao vrednost). Za prvu listu smo postavili listu pref_syncConnectionTypes_entries, a za drugu pref_syncConnectionTypes_values. Te dve liste smo definisali u datoteci arrays.xml unutar direktorijuma res/values/ (slika 3). Na slici 5 se vidi kako to izgleda na primeru, kada korisnik klikne na Sync interval.

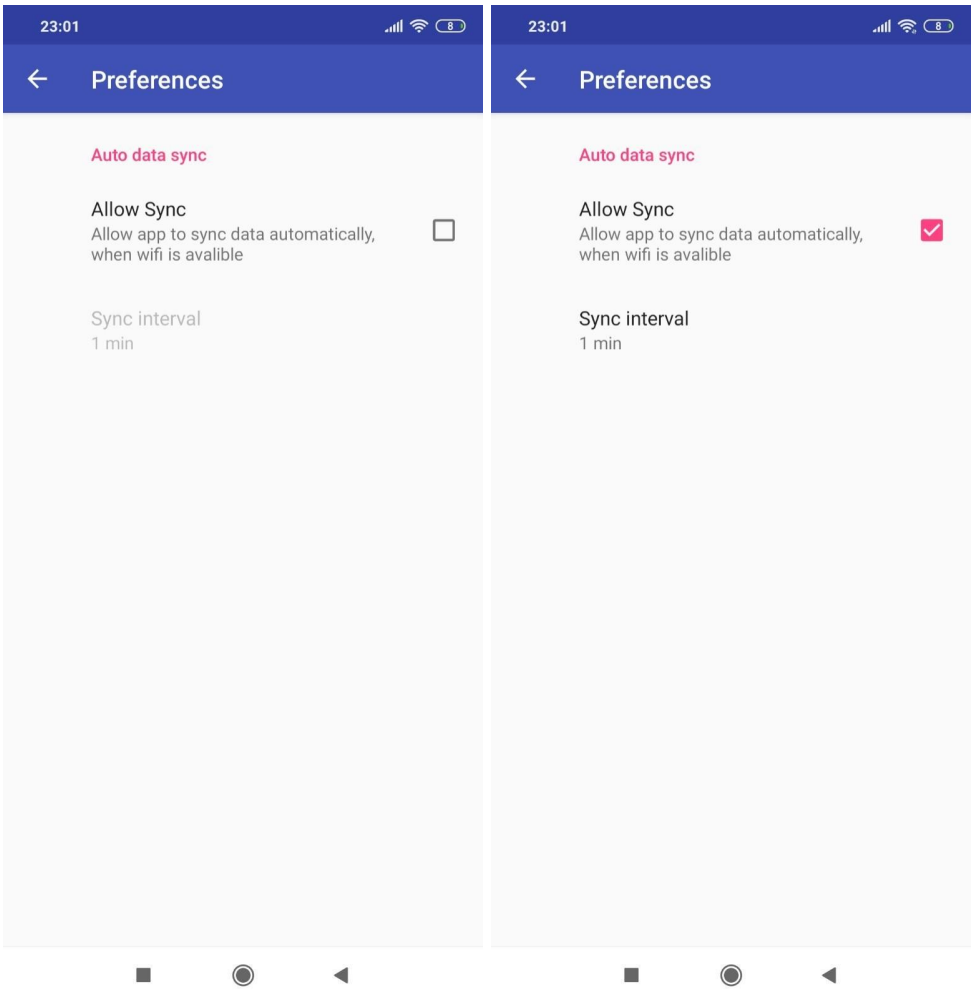
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

    <PreferenceCategory
        android:title="Auto data sync"
        android:key="sync_settings">

        <CheckBoxPreference
            android:defaultValue="false"
            android:key="pref_sync"
            android:summary="Allow app to sync data automatically, when wifi is aval..."
            android:title="Allow Sync" />

        <ListPreference
            android:dependency="pref_sync"
            android:dialogTitle="Sync interval"
            android:entries="@array/pref_syncConnectionTypes_entries"
            android:entryValues="@array/pref_syncConnectionTypes_values"
            android:key="pref_sync_list"
            android:title="Sync interval"
            android:summary="%s"
            android:defaultValue="1"/>

    </PreferenceCategory>

</PreferenceScreen>
```

Slika 2. *preferences.xml*

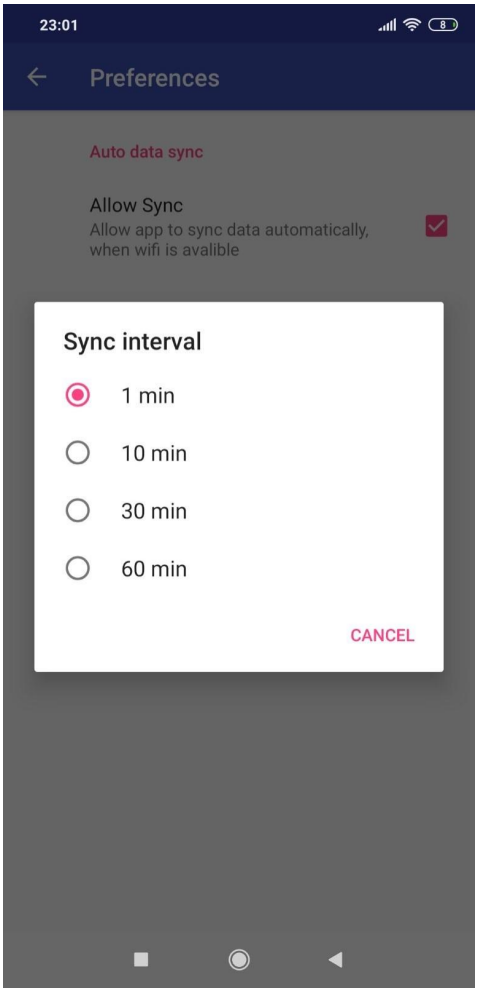


```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="pref_syncConnectionTypes_entries">
        <item>1 min</item>
        <item>10 min</item>
        <item>30 min</item>
        <item>60 min</item>
    </string-array>

    <string-array name="pref_syncConnectionTypes_values">
        <item>1</item>
        <item>10</item>
        <item>30</item>
        <item>60</item>
    </string-array>
```

Slika 3. Primeri nizova koji se prikazuju



Slika 4. Prikaz podešavanja

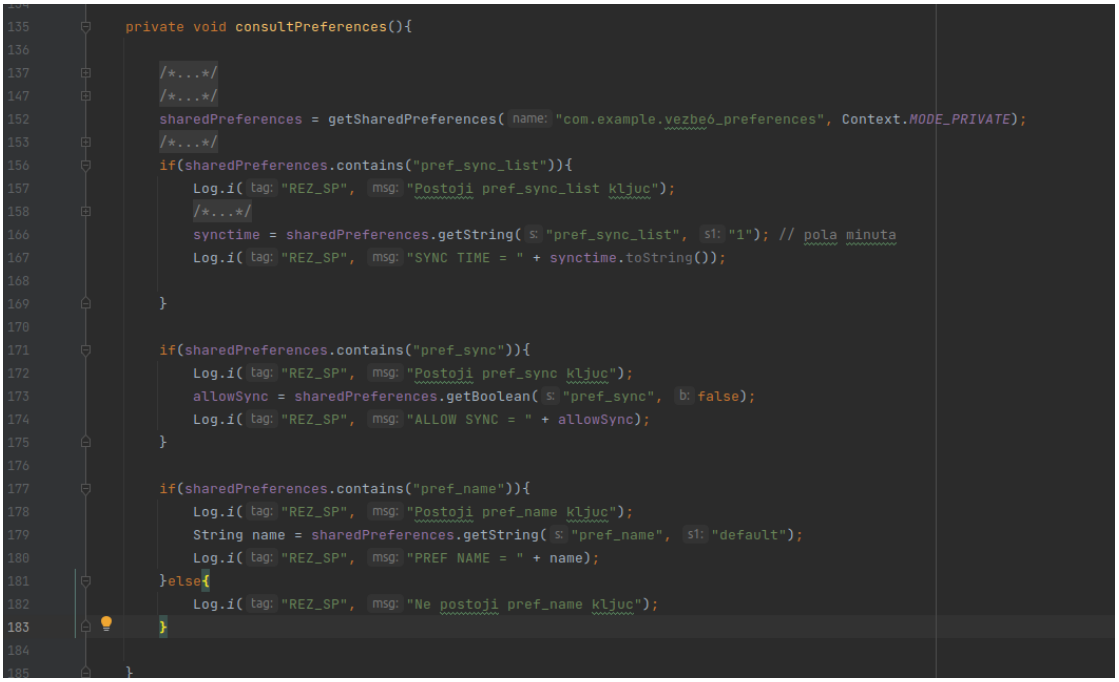


Slika 5. Prikaz liste sa opcijama koja se prikazuje kada se klikne na *Sync interval*

U *MainActivity* klasi se nalazi metoda *consultPreferences()*. U ovoj metodi pristupamo deljenim podešavanjima tako što pozivamo metodu *getSharedPreferences* ugrađene klase *Context* (slika 6).

Metoda *contains* proverava da li postoji podešavanje pod prosleđenim naslovom u okviru Shared Preferences. Metoda *getString* vraća string iz podešavanja, a metoda *getBoolean* vraća boolean vrednost. Gore smo napomenuli da u deljena podešavanja smeštamo parove (ključ, vrednost), te ćemo ključeve koristiti da dobavimo određene vrednosti. Kada pozivamo metode za dobavljanje, prosleđujemo 2 parametra:

- ključ - prvi parametar je ključ, čiju vrednost želimo da dobavimo.
- podrazumevana vrednost - drugi parametar je vrednost, koja treba da nam se vrati nazad, u slučaju da ništa nije zapisano pod prosleđenim ključem.



Slika 6. Čitanje zapisanih vrednosti

Na sličan način možemo da pristupimo deljenim podešavanjima i da smestimo neko podešavanje unutar njega. Pomoću metode *putString, putBoolean itd.* možemo da dodamo podešavanje sa nekim ključem i vrednošću (slika 7).

Slika 7. Zapisivanje vrednosti u deljenim podešavanjima

2. Rad sa datotekama

Podaci koje 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 isti adresni prostor). Najjednostavniji način da se prevaziđu ova ograničenja je korišćenjem datoteka.

Za rad sa datotekama koriste se klase iz java.io paketa, kao i metode klase Context koje olakšavaju pristup internom i/ili eksternom skladištu podataka, rad sa privremenim datotekama ili upravljanje pravima pristupa.

Skladišta podataka u Android uređaju:

- interno skladište - nalazi se u mobilnom uređaju, uvek dostupno, manjeg kapaciteta (nije proširivo), privatno.
- eksterno skladište - obično se nalazi na SD kartici, nije uvek dostupno, obično većeg kapaciteta (proširivo), javno.

Metode koje se koriste za pristupanje skladištima:

- 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 skladistu
- File getFilesDir() //privatan folder - kome samo tekuća aplikacija ima pristup
- File getExternalFilesDir(String type) // folder kome i druge aplikacije mogu pristupiti

Privremene datoteke treba skladištiti u cache direktorijumu (Android ih automatski briše kada ponestane slobodnog prostora).

Datoteke koje deli više aplikacija treba smestiti u javni eksterni direktorijum.

Pri korišćenju eksternog skladišta potrebno je obezbediti statičke (putem manifest fajla, slika 8) i dinamičke (putem dijaloga) permisije.

```
14
15      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
16      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
17
```

Slika 8. Statičke permisije

3. Snimanje fotografije i video zapisa

Snimanje fotografija i video zapisa je moguće korišćenjem postojeće sistemske aplikacije (*Camera*). Za snimanje fotografija i video snimaka koristimo *MediaRecorder*. MediaRecorder je API za snimanje različitih audio i video formata.

Potrebno je izvršiti sledeće korake:

- Omogućiti statička i dinamička prava pristupa.
- Detektovati kameru.
- Pristupiti kameri.
- Napraviti *Preview* klasu.
- Napraviti *Preview* raspored.
- Podesiti obradivače dogadaja.
- Snimiti fotografiju ili video u datoteku.
- Osloboditi kameru.

U *AndroidManifest* datoteci dodajemo novo pravo pristupa (CAMERA) slika 9.

```
11
12      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
13      <uses-permission android:name="android.permission.CAMERA" />
14
```

Slika 9. Definisanje statičkih prava pristupa za kameru

U metodi *onCreateView* (slika 10), klase *PictureCaptureFragment*, postavljamo naš *layout fragment_picture_capture.xml* i dobavljamo elemente sa *layout*-a. Potom vršimo proveru da li su permisije odobrene i ako nisu tražimo da se odobre.

```java
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_picture_capture, container,  attachToRoot: false);
    takePictureButton = view.findViewById(R.id.button_image);
    imageView = view.findViewById(R.id.imageview);

    startActivityForResult = registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(),
            result -> {
                if (result.getResultCode() == AppCompatActivity.RESULT_OK) {
                    Intent data = result.getData();
                    imageView.setImageURI(file);
                }
            }
    );
    if (ContextCompat.checkSelfPermission(getActivity(), Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
        takePictureButton.setEnabled(false);
        ActivityCompat.requestPermissions(getActivity(), new String[] { Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE },  requestCode: 0);
    }
    takePictureButton.setOnClickListener(view1 -> {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

        file = FileProvider.getUriForFile(getActivity(), GenericFileProvider.MY_PROVIDER, getOutputMediaFile());
        intent.putExtra(MediaStore.EXTRA_OUTPUT, file);
        startActivityForResult.launch(intent);
    });
    return view;
```

Slika 10. Metoda *onCreateView*

Kada korisnik želi da uslika potrebno je da klikne na dugme *Take a picture,* nakon čega će se pozvati setOnClickListener (slika 11). U toj metodi instanciramo nameru, startujemo aktivnost i primamo rezultat od aktivnosti.

GenericFileProvider je klasa koju smo napravili i ona nasleđuje *FileProvider*.

EXTRA_OUTPUT je URI koji određuje putanju i ime datoteke u koju će se sačuvati fotografija (ili video).

Ovo je primer dobijanja slike u punoj rezoluciji kamere.

```java
    }
    takePictureButton.setOnClickListener(view1 -> {
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);

        file = FileProvider.getUriForFile(getActivity(), GenericFileProvider.MY_PROVIDER, getOutputMediaFile());
        intent.putExtra(MediaStore.EXTRA_OUTPUT, file);
        startActivityForResult.launch(intent);
    });
```

Slika 11. Metoda *setOnClickListener*

Android *Camera* aplikacija može da sačuva fotografije ako joj postavimo putanju na kojoj treba da ih čuva. Uglavnom se fotografije, koje korisnik napravi, čuvaju na uređaju u javnom eksternom skladištu, tako da svaka aplikacija može da im pristupi. U svrhu generisanja putanje, napravljena je pomoćna metoda *getOutputMediaFile.*

Odgovarajući direktorijum za smeštanje deljenih fotografija se dobija pozivanjem metode *getExternalStoragePublicDirectory* i prosleđivanjem konstante DIRECTORY_PICTURES (slika 10).

Pošto naziv datoteke (putanja) mora biti jedinstven, na naziv direktorijuma dodajemo datum i vreme kada je fotografija napravljena.

```
67 @    private static File getOutputMediaFile(){
68          File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(
69                  Environment.DIRECTORY_PICTURES), child: "CameraDemo");
70
71          if (!mediaStorageDir.exists()){
72              if (!mediaStorageDir.mkdirs()){
73                  return null;
74              }
75          }
76
77          String timeStamp = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss").format(new Date());
78          return new File( pathname: mediaStorageDir.getPath() + File.separator +
79                  "IMG_"+ timeStamp + ".jpg");
80      }
```

Slika 10. Pomoćna metoda *getOutputMediaFile*

U *AndroidManifest* datoteci definisali smo i *FileProvider*, a prethodno i neophodna prava pristupa WRITE_EXTERNAL_STORAGE.

```
<provider
    android:name="rs.reviewer.GenericFileProvider"
    android:authorities="rs.reviewer.provider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/provider_paths"/>
</provider>
```

Slika 11. Definisanje *FileProvider*-a u *AndroidManifest* datoteci

5. Domaći

Domaći se nalazi na *Canvas-u (canvas.ftn.uns.ac.rs)* na putanji *Vežbe/08 Zadatak.pdf*

Primer *Vezbe8* možete preuzeti na sledećem linku:
https://gitlab.com/antesevicceca/mobilne-aplikacije-sit

Za dodatna pitanja možete se obratiti asistentima:

- Svetlana Antešević (svetlanaantesevic@uns.ac.rs)
- Jelena Matković (matkovic.jelena@uns.ac.rs)