Multimedija

Mobilne aplikacije

Stevan Gostojić

Fakultet tehničkih nauka, Novi Sad

17. decembar 2024.

Pregled sadržaja

Reprodukcija zvuka i videa

- Android platforma omogućava reprodukciju audio i video sadržaja u različitim formatima (i preko različitih mrežnih protokola).
- Moguće je reprodukovati sadržaj iz datoteka koje su skladištene kao (sirovi) resursi, datoteka u internom ili eksternom skladištu podataka ili iz toka podataka koji stiže preko mreže.

Reprodukcija zvuka i videa

Klasa	Opis
`MediaPlayer`	API za reprodukciju audio i video sadržaja.
`AudioManager`	Upravlja audio izvorima i audio izlazom.

Table 1: Ključne klase za reprodukciju multimedijalnog sadržaja.

AndroidManifest.xml

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

ExampleActivity.java

```
1  public class ExampleActivity extends Activity {
2
3    MediaPlayer mediaPlayer = null;
4
5    public void onStart() {
6      mediaPlayer = MediaPlayer.create(context, R.raw.sound_file);
7      mediaPlayer.start(); // No need to call prepare(); create() does that for
       you.
8    }
9
10   public void onStop() {
11     mediaPlayer.stop();
12     mediaPlayer.release();
13     mediaPlayer = null;
14   }
15 }
16
```

ExampleActivity.java

```
1  public class ExampleActivity extends Activity {
2
3    MediaPlayer mediaPlayer = null;
4
5    public void onStart() {
6      Uri uri = "file:/..."; // Initialize URI here
7      mediaPlayer = new MediaPlayer();
8      mediaPlayer.setDataSource(getApplicationContext(), uri);
9      mediaPlayer.prepare();
10     mediaPlayer.start();
11   }
12
13   public void onStop() {
14     mediaPlayer.stop();
15     mediaPlayer.release();
16     mediaPlayer = null;
17   }
18 }
19
```

ExampleActivity.java

```
1  public class ExampleActivity extends Activity {
2
3    MediaPlayer mediaPlayer = null;
4
5    public void onStart() {
6      String url = "http://........"; // Initialize URL here
7      mediaPlayer = new MediaPlayer();
8      mediaPlayer.setDataSource(url);
9      mediaPlayer.prepare(); // Might take long! (for buffering,
       etc.)
10     mediaPlayer.start();
11   }
12
13   public void onStop() {
14     mediaPlayer.stop();
15     mediaPlayer.release();
16     mediaPlayer = null;
17   }
18 }
19
```

Reprodukcija zvuka i videa

- Poziv `prepare` metode je (potencijalno) blokirajuća operacija.
- Ovu metodu treba pozvati u pozadinskoj niti ili umesto nje koristiti `prepareAsync` metodu i `MediaPlayer.OnPreparedListener` i `MediaPlayer.OnErrorListener` obrađivače događaja.

Dijagram stanja MediaPlayer-a

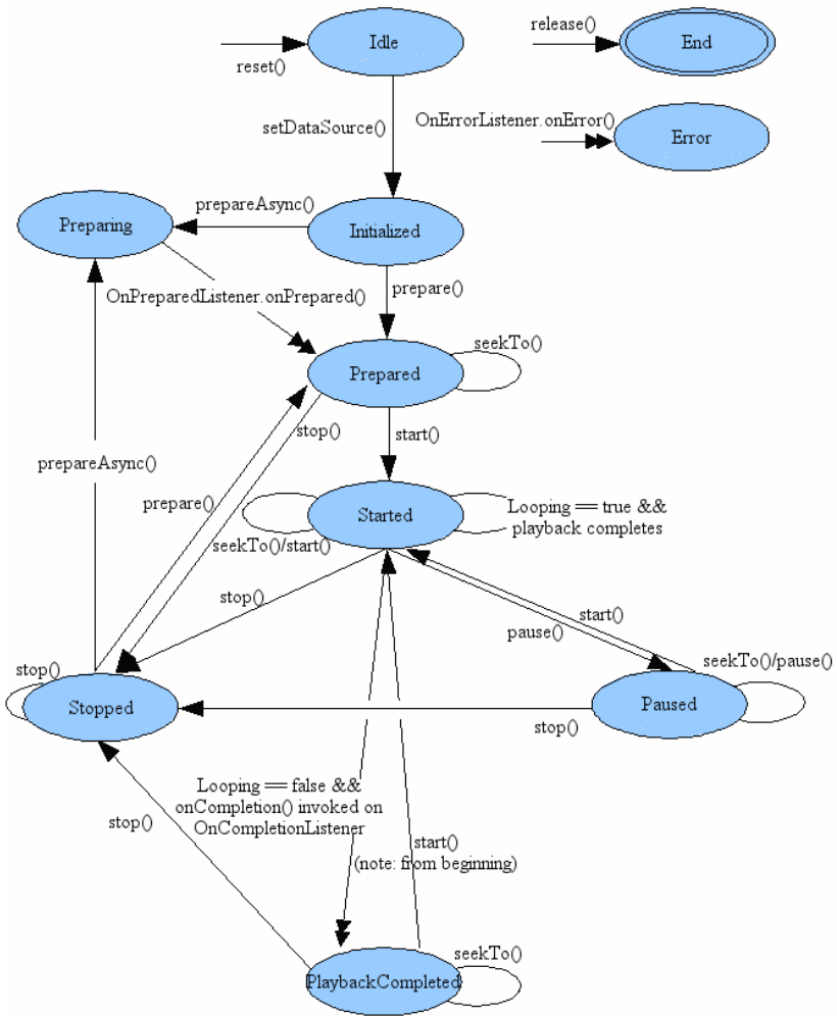

Figure 1: Dijagram stanja MediaPlayer-a.

ExampleService.java

```
1  public class ExampleService extends Service
2    implements MediaPlayer.OnPreparedListener, MediaPlayer.OnErrorListener {
3
4    private static final String ACTION_PLAY = "com.example.action.PLAY";
5
6    MediaPlayer mediaPlayer = null;
7
8    @Override
9    public int onStartCommand(Intent intent, int flags, int startId) {
10     if (intent.getAction().equals(ACTION_PLAY)) {
11       mediaPlayer = ... // Initialize it here
12       mediaPlayer.setOnPreparedListener(this);
13       mediaPlayer.setOnErrorListener(this);
14       mediaPlayer.prepareAsync(); // Prepare async to not block main thread
15     }
16   }
17
18   @Override
19   public void onDestroy() {
20     mediaPlayer.stop();
21     mediaPlayer.release();
22     mediaPlayer = null;
23   }
24
```

ExampleService.java

```
1    @Override
2    public void onPrepared(MediaPlayer mediaPlayer) {
3      mediaPlayer.start();
4    }
5
6    @Override
7    public boolean onError(MediaPlayer mediaPlayer, int what, int extra) {
8      // React appropriately...
9      // The MediaPlayer has moved to the Error state, must be reset!
10     mediaPlayer.reset();
11   }
12 }
13
```

Audio fokus

- Više audio izvora se može reprodukovati istovremeno (Android je multi-tasking platforma), ali postoji samo jedan audio izlaz.
- Audio fokus je mehanizam koji omogućava aplikacijama da se sinhronizuju prilikom korišćenja audio izlaza.
- On je kooperativan (od aplikacija se očekuje da se povinuju pravilima, ali se ta pravila ne nameću od strane sistema):
  - Kada aplikacija želi da reprodukuje zvuk, treba da zatraži audio fokus.
  - Kada aplikacija dobije audio fokus, može slobodno da reprodukuje zvuk (ali treba da "osluškuje" promenu audio fokusa).
  - Kada aplikacija izgubi audio fokus, treba da odmah zaustavi reprodukciju zvuka ili da smanji njegovu jačinu.

Audio fokus

Konstanta	Opis
`AUDIOFOCUS_GAIN`	Aplikacija je dobila audio fokus.
`AUDIOFOCUS_LOSS`	Aplikacija je na duži period izgubila audio fokus.
`AUDIOFOCUS_LOSS_TRANSIENT`	Aplikacija je privremeno izgubila audio fokus.
`AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK`	Aplikacija je privremeno izgubila audio fokus (ali može tiho da reprodukuje zvuk).

Table 2: Stanja audio fokusa.

ExampleService.java

```
1 AudioManager audioManager = (AudioManager) getSystemService(Context.
      AUDIO_SERVICE);
2 int result = audioManager.requestAudioFocus(
3   new AudioFocusRequest.Builder(AudioManager.AUDIOFOCUS_GAIN).build());
4
5 if (result != AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
6   // could not get audio focus.
7 }
8
```

ExampleService.java

```
1  class MyService extends Service
2    implements AudioManager.OnAudioFocusChangeListener {
3
4    public void onAudioFocusChange(int focusChange) {
5      switch (focusChange) {
6      case AudioManager.AUDIOFOCUS_GAIN:
7        // Resume playback
8        if (mediaPlayer == null) initMediaPlayer();
9        else if (!mediaPlayer.isPlaying()) mediaPlayer.start();
10       mediaPlayer.setVolume(1.0f, 1.0f);
11       break;
12     case AudioManager.AUDIOFOCUS_LOSS:
13       // Lost focus for an unbounded amount of time: stop playback and release
       media player
14       if (mediaPlayer.isPlaying()) mediaPlayer.stop();
15       mediaPlayer.release();
16       mediaPlayer = null;
17       break;
18     case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT:
19       // Lost focus for a short time, but we have to stop
20       // playback. We don't release the media player because playback
21       // is likely to resume
22       if (mediaPlayer.isPlaying()) mediaPlayer.pause();
23       break;
24     case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK:
25       // Lost focus for a short time, but it's ok to keep playing
26       // at an attenuated level
27       if (mediaPlayer.isPlaying()) mediaPlayer.setVolume(0.1f, 0.1f);
28       break;
29     }
30   }
31 }
32
```

Reprodukcija videa

Da bi se reprodukovao video sadržaj potrebno je deklarisati pogled koji će prikazati sadržaj:

- `VideoView`
- `SurfaceView`

i definisati funkciju koja ga reprodukuje.

main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:orientation="horizontal"
5     android:layout_width="fill_parent"
6     android:layout_height="fill_parent">
7
8    <VideoView
9       android:id="@+id/videoview"
10      android:layout_height="fill_parent"
11      android:layout_width="fill_parent" />
12
13 </LinearLayout>
14
```

ExampleActivity.java

```
1  public void playVideo() {
2    VideoView videoView = (VideoView) findViewById(R.id.videoview);
3    videoView.setVideoPath("/sdcard/test2.3gp");
4    //videoView.setVideoURI("http://www.mysite.com/videos/myvideo.3
       gp");
5    if (videoView.canSeekForward())
6      videoView.seekTo(videoView.getDuration()/2);
7    videoView.start();
8    // Do something ...
9    videoView.stopPlayback();
10 }
11
```

Pregled sadržaja

Snimanje zvuka

- Android platforma omogućava snimanje zvuka posredstvom mikrofona.
- Snimanje zvuka je moguće samo na fizičkim uređajima; emulator nema tu mogućnost.

Snimanje zvuka

Klasa	Opis
`MediaRecorder`	API za snimanje zvuka.

Table 3: Ključna klasa `MediaRecorder` API-a.

Snimanje zvuka

Za snimanje zvuka je potrebno izvršiti sledeće korake:

- Instancirati `MediaRecorder`.
- Postaviti audio izvor korišćenjem `setAudioSource` metode (verovatno na `MediaRecorder.AudioSource.MIC`).
- Postaviti izlazni format korišćenjem `setOutputFormat` metode.
- Postaviti ime izlazne datoteke korišćenjem `setOutputFile` metode.
- Postaviti audio koder korišćenjem `setAudioEncoder` metode.

Snimanje zvuka

- Pozvati `prepare` metodu.
- Pozvati `start` metodu za početak snimanja zvuka.
- Pozvati `stop` metodu za završetak snimanja zvuka.
- Osloboditi instancu `MediaRecorder`-a pozivanjem `release` metode.

Dijagram stanja MediaPlayer-a

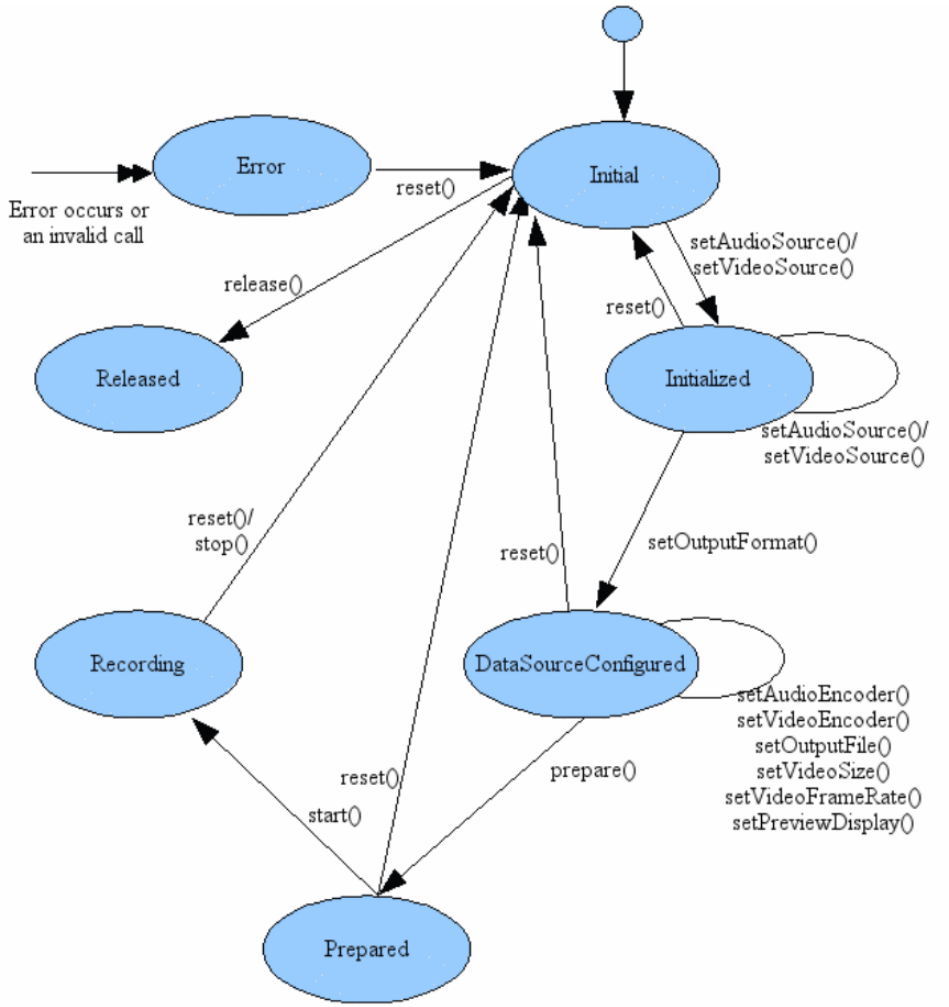

Figure 2: Dijagram stanja `MediaRecorder`-a.

ExampleService.java

```
1  private void startRecording() {
2    recorder = new MediaRecorder(context);
3    recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
4    recorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
5    recorder.setOutputFile(mFileName);
6    recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
7
8    try {
9      recorder.prepare();
10   } catch (IOException e) {
11     Log.e(LOG_TAG, "prepare() failed");
12   }
13
14   recorder.start();
15 }
16
```

ExampleService.java

```
1 private void stopRecording() {
2   recorder.stop();
3   recorder.release();
4   recorder = null;
5 }
6
```

Pregled sadržaja

Snimanje fotografija i videa

Snimanje fotografija i videa je moguće korišćenjem:

- postojeće aplikacije (tj. aktivnosti)
- `android.hardware.camera2` paketa

Korišćenje postojeće aplikacije

- Instancirati `Intent`.
- Startovati aktivnost.
- Primiti rezultat od aktivnosti.

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest ... >
3    <uses-permission android:name="android.permission.CAMERA" />
4    <uses-permission android:name="android.permission.RECORD_AUDIO" />
5    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
6    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7
8    <uses-feature android:name="android.hardware.camera" android:required="false"
       />
9  </manifest>
10
```

ExampleActivity.java

```
1  private void captureImage() {
2    // Create Intent to take a picture and return control to the calling
       application
3    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
4
5    // Create a file to save the image
6    fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE);
7    // Set the image file name
8    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
9
10   // Start the image capture Intent
11   startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
12 }
13
```

ExampleActivity.java

```
1  private void captureVideo() {
2    // Create new Intent
3    Intent intent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
4
5    // Create a file to save the video
6    fileUri = getOutputMediaFileUri(MEDIA_TYPE_VIDEO);
7    // Set the image file name
8    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
9
10   // Set the video image quality to high
11   intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);
12
13   // Start the Video Capture Intent
14   startActivityForResult(intent, CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE);
15 }
16
```

ExampleActivity.java

```
1  @Override
2  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
3    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
4      if (resultCode == RESULT_OK) {
5        // Image captured and saved to fileUri specified in the Intent
6        Toast.makeText(this, "Image saved to:\n" + data.getData(), Toast.LENGTH_LONG).show();
7      } else if (resultCode == RESULT_CANCELED) {
8        // User cancelled the image capture
9      } else {
10       // Image capture failed, advise user
11     }
12   }
13
14   if (requestCode == CAPTURE_VIDEO_ACTIVITY_REQUEST_CODE) {
15     if (resultCode == RESULT_OK) {
16       // Video captured and saved to fileUri specified in the Intent
17       Toast.makeText(this, "Video saved to:\n" + data.getData(), Toast.LENGTH_LONG).show();
18     } else if (resultCode == RESULT_CANCELED) {
19       // User cancelled the video capture
20     } else {
21       // Video capture failed, advise user
22     }
23   }
24 }
25
```

Korišćenje postojeće aplikacije

Konstanta	Opis
`EXTRA_OUTPUT`	URI koji određuje putanju i ime datoteke u koju će se sačuvati fotografija ili video.
`EXTRA_VIDEO_QUALITY`	0 (za najniži kvalitet i najmanju veličinu datoteke) ili 1 (za najviši kvalitet i najveću veličinu datoteke).
`EXTRA_DURATION_LIMIT`	Ograničenje dužine videa koji se snima (u sekundama).
`EXTRA_SIZE_LIMIT`	Ograničenje veličine datoteke (u bajtovima).

Table 4: Ekstra parametri `Intent`-a.

Korišćenje `MediaRecorder` API-a

- Ukoliko je potrebno prilagoditi izgled ili funkciju aktivnosti za snimanje fotografija ili videa, treba koristiti API iz `android.hardware.camera2` paketa.

Korišćenje android.hardware.camera2 paketa

Klasa	Opis
CameraManager	servis za upravljanje kamerama.
CameraDevice	reprezentuje jednu kameru.
CameraCharacteristics	karakteristike kamere.
CameraCaptureSession	sesija za komunikaciju sa kamerom.
CaptureRequest	zahtev koji se upućuje kameri putem sesije.

Table 5: Ključne klase za pristup kameri.

Korišćenje `android.hardware.camera2` paketa

Za snimanje fotografija i videa korišćenjem `android.hardware.camera2` paketa potrebno je izvršiti sledeće korake:

- Pribavljanje permisija
- Izbor kamere
- Otvaranje konekcije ka kameri
- Izbor površine tj. Surface (npr. SurfaceView na layout-u ili ImageView za čuvanje fotografija ili MediaRecorder za čuvanje video zapisa
- Kreiranje capture sesije
- Kreiranje capture zahteva
- Postavljanje zahteva u sesiju

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest ... >
3    <uses-permission android:name="android.permission.CAMERA" />
4    <uses-permission android:name="android.permission.RECORD_AUDIO" />
5    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
6    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
7    <!-- runtime permissions should also be gained -->
8
9    <uses-feature android:name="android.hardware.camera" />
10
11   <activity android:name=".ExampleActivity" ...>
12     <intent-filter>
13       <action android:name="android.intent.action.MAIN" />
14       <category android:name="android.intent.category.LAUNCHER" />
15     </intent-filter>
16   </activity>
17 </manifest>
18
```

activity_example.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:id="@+id/main"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      tools:context=".ExampleActivity">
10
11     <SurfaceView
12         android:id="@+id/surfaceView"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content" />
15
16 </LinearLayout>
17
```

ExampleActivity.java

```
1  private boolean checkCameraHardware(Context context) {
2    if (context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA)
       ){
3      // This device has a camera
4      return true;
5    } else {
6      // No camera on this device
7      return false;
8    }
9  }
10
```

ExampleActivity.java

```
1  import android.hardware.camera2.CameraManager;
2  import android.hardware.camera2.CameraCharacteristics;
3  // ...
4  private static CameraManager cameraManager;
5
6  @Override
7  protected void onCreate(Bundle savedInstanceState) {
8    super.onCreate(savedInstanceState);
9    setContentView(R.layout.activity_main);
10
11   cameraManager = (CameraManager) getApplicationContext()
12     .getSystemService(Context.CAMERA_SERVICE);
13
14   String cameraId = null;
15   try {
16     for (String id : cameraManager.getCameraIdList()) {
17       CameraCharacteristics characteristics =
18             cameraManager.getCameraCharacteristics(id);
19       if (characteristics.get(CameraCharacteristics.LENS_FACING) ==
20             CameraCharacteristics.LENS_FACING_FRONT)
21         cameraId = id;
22     }
23   } catch (CameraAccessException e) {
24     e.printStackTrace();
25   }
26   useCamera(cameraId);
27 }
28 // CameraCharacteristics.LENS_FACING_BACK refers to the back camera
```

ExampleActivity.java

```
1  private void useCamera(String cameraId) {
2    try {
3      cameraManager.openCamera(cameraId, new CameraDevice.StateCallback() {
4        @Override
5        public void onOpened(@NonNull CameraDevice cameraDevice) {
6          SurfaceView sv = findViewById(R.id.surfaceView);
7          List<Surface> surfaces = new ArrayList<>();
8          surfaces.add(sv.getHolder().getSurface());
9          try {
10           cameraDevice.createCaptureSession(surfaces,
11            new CameraCaptureSession.StateCallback() {
12             @Override
13             public void onConfigured(@NonNull CameraCaptureSession session) {
14               // when the session is configured, capture requests can be initiated
15               initiateRequests(session);  // shown on the next slide
16             }
17             @Override
18             public void onConfigureFailed(@NonNull CameraCaptureSession session) {...}
19           }, null);
20         } catch (CameraAccessException e) {
21           e.printStackTrace();
22         }
23       }
24       @Override
25       public void onDisconnected(@NonNull CameraDevice cameraDevice) { ... }
26       @Override
27       public void onError(@NonNull CameraDevice cameraDevice, int i) { ... }
28     }, new Handler());
29   } catch (CameraAccessException e) {
30     throw new RuntimeException(e);
31   }
32 }
```

ExampleActivity.java (video preview)

```
1  private void initiateRequests(CameraCaptureSession session) {
2    try {
3      CaptureRequest.Builder captureRequestBuilder =
4        session.getDevice().createCaptureRequest(CameraDevice.TEMPLATE_PREVIEW);
5
6      captureRequestBuilder.addTarget(sv.getHolder().getSurface());
7
8      session.setRepeatingRequest(captureRequestBuilder.build(), null, null);
9
10   } catch (CameraAccessException ex) {
11     ex.printStackTrace();
12   } catch (Exception e) {
13     e.printStackTrace();
14   }
15 }
```

Metoda createCaptureRequest(templateType)

Konstanta	Opis
TEMPLATE_PREVIEW	Režim prilagođen prikazivanju (frame rate ima prioritet u odnosu na kvalitet).
TEMPLATE_STILL_CAPTURE	Režim fotografisanja (prioritet je na kvalitetu slike).
TEMPLATE_RECORD	Režim snimanja videa (održava stabilan frame rate).
TEMPLATE_VIDEO_SNAPSHOT	Režim snimanja slika tokom snimanja videa (kvalitet slike je prioritet dokle god se time ne ugrožava snimanje videa).
TEMPLATE_ZERO_SHUTTER_LAG	Režim snimanja slika bez kašnjenja (prioritet je na kvalitetu dokle god to ne ugrožava frame rate prikaza).
TEMPLATE_MANUAL	Onemogućava automatska podešavanja (fokus, ekspozicija, nijansa bele boje), kako bi se ove vrednosti programski kontrolisale.

Table 6: Vrednosti parametra templateType u metodi createCaptureRequest

Metode klase CameraCaptureSession

Konstanta	Opis
capture	Zahtev za snimanje pojedinačne slike.
setRepeatingRequest	Zahtev koji se uzastopno ponavlja.
captureBurst	Sekvenca višestrukih zahteva koji se trebaju uzastopno izvršiti.
setRepeatingBurst	Sekvenca višestrukih zahteva koji se uzastopno ponavljaju.
stopRepeating	Obustavlja sve ponavljajuće (repeating) zahteve.
close	Zatvaranje sesije.

Table 7: Neke od metoda klase CameraCaptureSession

ExampleActivity.java (video recording)

```java
private void recordVideo(CameraCaptureSession session) {
  try {
    MediaRecorder mediaRecorder = new MediaRecorder();
    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    mediaRecorder.setVideoSource(MediaRecorder.VideoSource.SURFACE);
    mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
    mediaRecorder.setOutputFile("file:/...");
    mediaRecorder.setVideoEncodingBitRate(bitsPerSecond);
    mediaRecorder.setVideoFrameRate(framerate);
    mediaRecorder.setVideoSize(width, height); // in accordance with camera characteristics
    mediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.H264);
    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
    mediaRecorder.prepare();

    CaptureRequest.Builder captureRequestBuilder =
      session.getDevice().createCaptureRequest(CameraDevice.TEMPLATE_RECORD);
    captureRequestBuilder.addTarget(mediaRecorder.getSurface());
    captureRequestBuilder.addTarget(sv.getHolder().getSurface());
    // start recording
    mediaRecorder.start();
    session.setRepeatingRequest(captureRequestBuilder.build(), null, null);
    // ...
    // stop recording
    mediaRecorder.stop();
    mediaRecorder.reset();
  } catch (CameraAccessException ex) {
    ex.printStackTrace();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

ExampleActivity.java (image capturing)

```
1  private void captureImage(CameraCaptureSession session) {
2    try {
3      ImageReader imageReader = ImageReader.newInstance(width, height, ImageFormat.JPEG, 1);
4      imageReader.setOnImageAvailableListener(new ImageReader.OnImageAvailableListener() {
5        @Override
6        public void onImageAvailable(ImageReader reader) {
7          Image image = null;
8          OutputStream output = null;
9          try {
10           image = imageReader.acquireLatestImage();
11           ByteBuffer buffer = image.getPlanes()[0].getBuffer();
12           byte[] bytes = new byte[buffer.capacity()];
13           buffer.get(bytes);
14           output = new FileOutputStream("file:/...");
15           output.write(bytes);
16         } catch (Exception e) {
17           e.printStackTrace();
18         } finally {
19           if (image != null) image.close();
20           if (output != null) output.close();
21         }
22       }
23     }, null);
24     CaptureRequest.Builder captureRequestBuilder =
25       session.getDevice().createCaptureRequest(CameraDevice.TEMPLATE_STILL_CAPTURE);
26     captureRequestBuilder.addTarget(imageReader.getSurface());
27     session.capture(captureRequestBuilder.build(), null, null);
28   } catch (CameraAccessException ex) {
29     ex.printStackTrace();
30   } catch (Exception e) {
31     e.printStackTrace();
32   }
33 }
```