Multimedija

Mobilne aplikacije

Stevan Gostojić

Fakultet tehničkih nauka, Novi Sad

20. decembar 2022.

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.setAudioStreamType(AudioManager.STREAM_MUSIC);
9      mediaPlayer.setDataSource(getApplicationContext(), uri);
10     mediaPlayer.prepare();
11     mediaPlayer.start();
12   }
13
14   public void onStop() {
15     mediaPlayer.stop();
16     mediaPlayer.release();
17     mediaPlayer = null;
18   }
19 }
20
```

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.setAudioStreamType(AudioManager.STREAM_MUSIC);
9      mediaPlayer.setDataSource(url);
10     mediaPlayer.prepare(); // Might take long! (for buffering,
       etc.)
11     mediaPlayer.start();
12   }
13
14   public void onStop() {
15     mediaPlayer.stop();
16     mediaPlayer.release();
17     mediaPlayer = null;
18   }
19 }
20
```

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

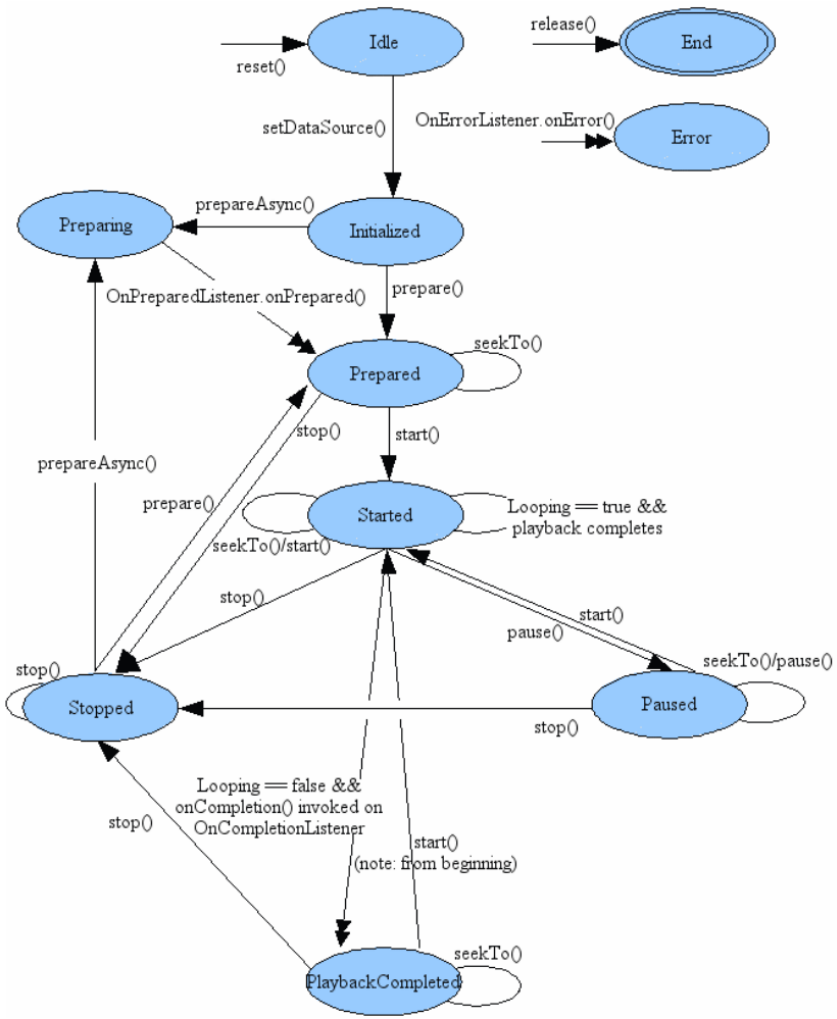

Figure 1: Dijagram stanja MediaPlayer-a.

ExampleService.java

```
1  public class ExampleService
2    extends Service
3    implements MediaPlayer.OnPreparedListener, MediaPlayer.OnErrorListener {
4
5    private static final String ACTION_PLAY = "com.example.action.PLAY";
6
7    MediaPlayer mediaPlayer = null;
8
9    @Override
10   public int onStartCommand(Intent intent, int flags, int startId) {
11     if (intent.getAction().equals(ACTION_PLAY)) {
12       mediaPlayer = ... // Initialize it here
13       mediaPlayer.setOnPreparedListener(this);
14       mediaPlayer.setOnErrorListener(this);
15       mediaPlayer.prepareAsync(); // Prepare async to not block main thread
16     }
17   }
18
19   @Override
20   public void onDestroy() {
21     mediaPlayer.stop();
22     mediaPlayer.release();
23     mediaPlayer = null;
24   }
25
```

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 sa 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(this, AudioManager.STREAM_MUSIC,
3   AudioManager.AUDIOFOCUS_GAIN);
4
5 if (result != AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
6   // could not get audio focus.
7 }
8
```

ExampleService.java

```
1  class MyService
2    extends Service
3    implements AudioManager.OnAudioFocusChangeListener {
4
5    public void onAudioFocusChange(int focusChange) {
6      switch (focusChange) {
7      case AudioManager.AUDIOFOCUS_GAIN:
8        // Resume playback
9        if (mediaPlayer == null) initMediaPlayer();
10       else if (!mediaPlayer.isPlaying()) mediaPlayer.start();
11       mediaPlayer.setVolume(1.0f, 1.0f);
12       break;
13     case AudioManager.AUDIOFOCUS_LOSS:
14       // Lost focus for an unbounded amount of time: stop playback and release
       media player
15       if (mediaPlayer.isPlaying()) mediaPlayer.stop();
16       mediaPlayer.release();
17       mediaPlayer = null;
18       break;
19     case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT:
20       // Lost focus for a short time, but we have to stop
21       // playback. We don't release the media player because playback
22       // is likely to resume
23       if (mediaPlayer.isPlaying()) mediaPlayer.pause();
24       break;
25     case AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK:
26       // Lost focus for a short time, but it's ok to keep playing
27       // at an attenuated level
28       if (mediaPlayer.isPlaying()) mediaPlayer.setVolume(0.1f, 0.1f);
29       break;
30     }
31   }
32 }
33
```

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.setKeepScreenOn(true);
4    videoView.setVideoPath("/sdcard/test2.3gp");
5    //videoView.setVideoUri("http://www.mysite.com/videos/myvideo.3
       gp");
6    if (videoView.canSeekForward())
7      videoView.seekTo(videoView.getDuration()/2);
8    videoView.start();
9    // Do something ...
10   videoView.stopPlayback();
11 }
12
```

Pregled sadržaja

Rutiranje multimedijalnih tokova

Android platforma omogućava korišćenje eksternih audio i video uređaja za reprodukciju multimedijalnog sadržaja u dva moda:

- Remote Playback (koristi eksterni uređaj za rukovanje pronalaženjem, dekodiranjem i reprodukcijom sadržaja; na primer Google Cast)
- Secondary Output (aplikacija pronalazi, dekodira i reprodukuje video ili audio direktno na eksterni uređaj).

Rutiranje multimedijalnih tokova

Figure 2: ActioBar.

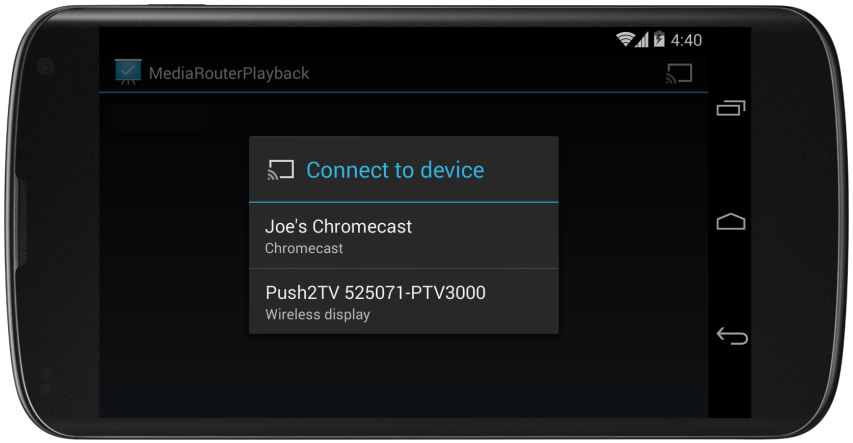

Figure 3: Media Router Selector UI.

Rutiranje multimedijalnih tokova

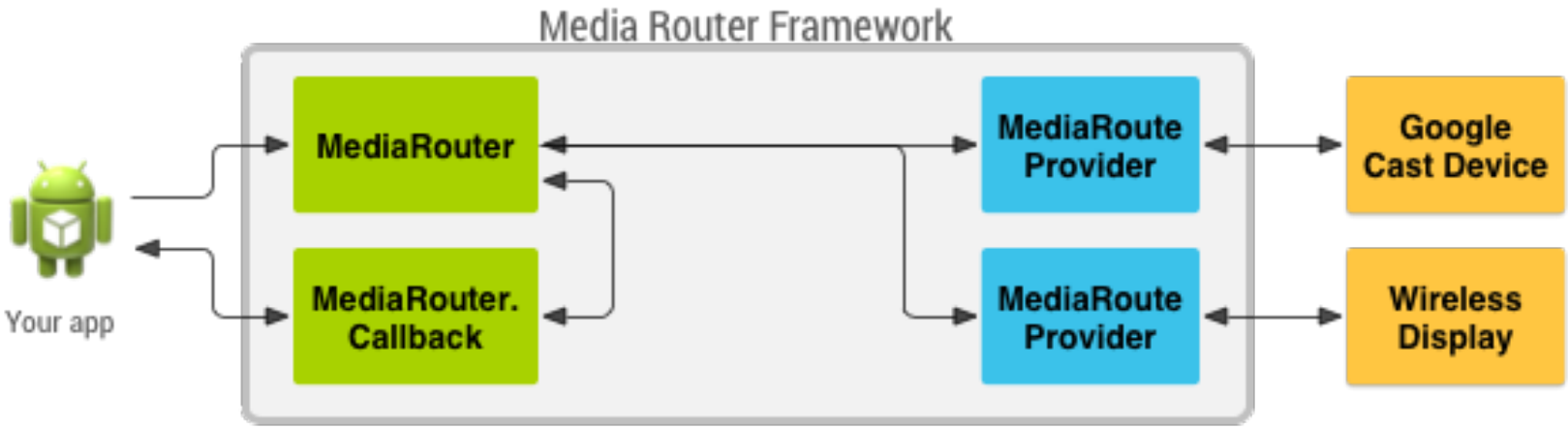

Figure 4: Arhitektura Media Router Framework-a.

Rutiranje multimedijalnih tokova

Klasa	Opis
`MediaRouter`	Omogućava rutiranje multimedijalnih tokova sa mobilnog uređaja na eksterne multimedijalne uređaje.
`MediaRouter.Callback`	Obrađuje događaje prouzrokovane promenom multimedijalnih ruta.
`MediaRouteSelector`	Opisuje mogućnosti multimedijalnih ruta.
`MediaRouteProvider`	Koristi se za objavljivanje multimedijalnih ruta.

Table 3: Ključne klase `Media Router Framework` API-a.

sample_media_router_menu.xml

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

    <item
        android:id="@+id/media_route_menu_item"
        android:title="@string/media_route_menu_title"
        app:actionProviderClass="android.support.v7.app.MediaRouteActionProvider"
        app:showAsAction="always" />

</menu>

```

MediaRouterPlaybackActivity.java

```
1  public class MediaRouterPlaybackActivity extends ActionBarActivity {
2
3    private MediaRouteSelector selector;
4
5    @Override
6    protected void onCreate(Bundle savedInstanceState) {
7      super.onCreate(savedInstanceState);
8      setContentView(R.layout.activity_main);
9
10     // Create a route selector for the type of routes your app supports.
11     selector = new MediaRouteSelector.Builder()
12       .addControlCategory(MediaControlIntent.CATEGORY_LIVE_AUDIO)
13       .addControlCategory(MediaControlIntent.CATEGORY_LIVE_VIDEO)
14       .addControlCategory(MediaControlIntent.CATEGORY_REMOTE_PLAYBACK)
15       .build();
16   }
17
```

MediaRouterPlaybackActivity.java

```
1    @Override
2    public boolean onCreateOptionsMenu(Menu menu) {
3      super.onCreateOptionsMenu(menu);
4      // Inflate the menu and configure the media router action provider.
5      getMenuInflater().inflate(R.menu.sample_media_router_menu, menu);
6      // Attach the MediaRouteSelector to the menu item
7      MenuItem mediaRouteMenuItem = menu.findItem(R.id.media_route_menu_item);
8      MediaRouteActionProvider mediaRouteActionProvider =
9        (MediaRouteActionProvider)MenuItemCompat.getActionProvider(
       mediaRouteMenuItem);
10     mediaRouteActionProvider.setRouteSelector(selector);
11
12     // Return true to show the menu.
13     return true;
14   }
15 }
16
```

MediaRouterPlaybackActivity.java

```
1  private final MediaRouter.Callback mediaRouterCallback = new MediaRouter.
       Callback() {
2
3    @Override
4    public void onRouteSelected(MediaRouter router, RouteInfo route) {
5      Log.d(TAG, "onRouteSelected: route=" + route);
6      if (route.supportsControlCategory(MediaControlIntent.
       CATEGORY_REMOTE_PLAYBACK)){
7        // Remote playback device
8        updateRemotePlayer(route);
9      } else {
10       // Secondary output device
11       updatePresentation(route);
12     }
13   }
14
```

MediaRouterPlaybackActivity.java

```
1   @Override
2   public void onRouteUnselected(MediaRouter router, RouteInfo route) {
3     Log.d(TAG, "onRouteUnselected: route=" + route);
4
5     if (route.supportsControlCategory(MediaControlIntent.
      CATEGORY_REMOTE_PLAYBACK)) {
6       // Remote playback device
7       updateRemotePlayer(route);
8     } else {
9       // Secondary output device
10      updatePresentation(route);
11    }
12  }
13
14  @Override
15  public void onRoutePresentationDisplayChanged(MediaRouter router, RouteInfo
      route) {
16    Log.d(TAG, "onRoutePresentationDisplayChanged: route=" + route);
17
18    if (route.supportsControlCategory(MediaControlIntent.
      CATEGORY_REMOTE_PLAYBACK)) {
19      // Remote playback device
20      updateRemotePlayer(route);
21    } else {
22      // Secondary output device
23      updatePresentation(route);
24    }
25  }
26 }
27
```

MediaRouterPlaybackActivity.java

```
1  private void updateRemotePlayer(RouteInfo route) {
2    // Changed route: tear down previous client
3    if (mRoute != null && mRemotePlaybackClient != null) {
4      mRemotePlaybackClient.release();
5      mRemotePlaybackClient = null;
6    }
7
8    // Save new route
9    mRoute = route;
10
11   // Attach new playback client
12   mRemotePlaybackClient = new RemotePlaybackClient(this, mRoute);
13
14   // Send file for playback
15   mRemotePlaybackClient.play(Uri.parse(
16     "http://archive.org/download/Sintel/sintel-2048-stereo_512kb.mp4"),
17     "video/mp4", null, 0, null, new ItemActionCallback() {
18       @Override
19       public void onResult(Bundle data, String sessionId, MediaSessionStatus
       sessionStatus,
20           String itemId, MediaItemStatus itemStatus) {
21         logStatus("play: succeeded for item " + itemId);
22       }
23
24       @Override
25       public void onError(String error, int code, Bundle data) {
26         logStatus("play: failed - error:"+ code +" - "+ error);
27       }
28   });
29 }
30
```

SamplePresentation.java

```
1  public class SamplePresentation extends Presentation {
2
3    public SamplePresentation(Context outerContext, Display display) {
4      super(outerContext, display);
5    }
6
7    @Override
8    protected void onCreate(Bundle savedInstanceState) {
9      super.onCreate(savedInstanceState);
10     // Notice that we get resources from the context of the Presentation
11     Resources resources = getContext().getResources();
12
13     // Inflate a layout.
14     setContentView(R.layout.presentation_with_media_router_content);
15
16     // Add presentation content here:
17     // Set up a surface view for visual interest
18     mSurfaceView = (GLSurfaceView)findViewById(R.id.surface_view);
19     mSurfaceView.setRenderer(new CubeRenderer(false));
20   }
21 }
22
```

SamplePresentation.java

```
1  private void updatePresentation(RouteInfo route) {
2    // Get its Display if a valid route has been selected
3    Display selectedDisplay = null;
4    if (route != null) {
5      selectedDisplay = route.getPresentationDisplay();
6    }
7
8    // Dismiss the current presentation if the display has changed or no new
9    // route has been selected
10   if (mPresentation != null && mPresentation.getDisplay() != selectedDisplay) {
11     mPresentation.dismiss();
12     mPresentation = null;
13   }
14
```

SamplePresentation.java

```
1   // Show a new presentation if the previous one has been dismissed and a
2   // route has been selected.
3   if (mPresentation == null && selectedDisplay != null) {
4     // Initialize a new Presentation for the Display
5     mPresentation = new SamplePresentation(this, selectedDisplay);
6     mPresentation.setOnDismissListener(new DialogInterface.OnDismissListener()
      {
7       // Listen for presentation dismissal and then remove it
8       @Override
9       public void onDismiss(DialogInterface dialog) {
10        if (dialog == mPresentation) {
11          mPresentation = null;
12        }
13      }
14    });
15
16    // Try to show the presentation, this might fail if the display has
17    // gone away in the meantime
18    try {
19      mPresentation.show();
20    } catch (WindowManager.InvalidDisplayException ex) {
21      // Couldn't show presentation - display was already removed
22      mPresentation = null;
23    }
24  }
25 }
26
```

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 i videa.

Table 4: 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

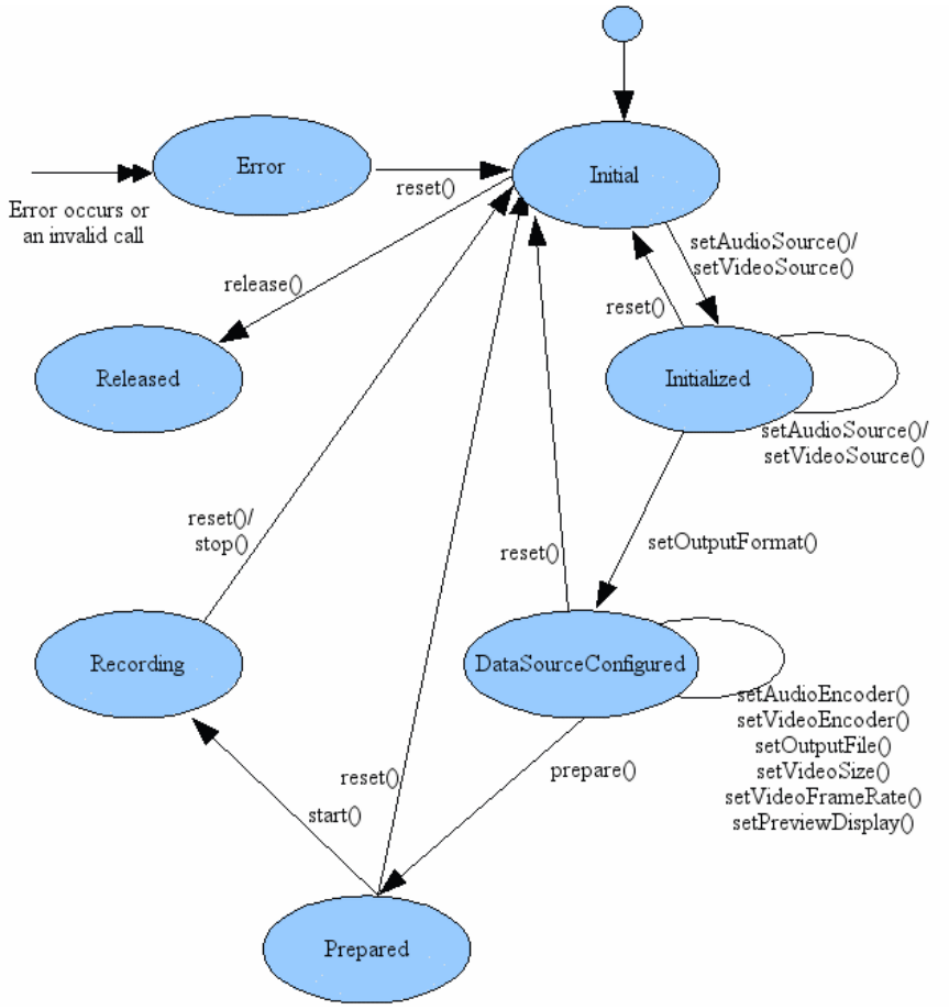

Figure 5: Dijagram stanja `MediaRecorder`-a.

ExampleService.java

```
1  private void startRecording() {
2    recorder = new MediaRecorder();
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)
- `MediaRecorder` API-a

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 5: Ekstra parametri `Intent`-a.

Korišćenje `MediaRecorder` API-a

- Ukoliko je potrebno prilagoditi izgled ili funkciju aktivnosti za snimanje fotografija ili videa, treba koristiti `MediaRecorder` API.

Korišćenje `MediaRecorder` API-a

Klasa	Opis
`MediaRecorder`	API za snimanje zvuka i videa.
`Camera`	API za upravljanje kamerama.
`SurfaceView`	Pogled za prikaz videa korisniku.

Table 6: Ključne klase za snimanje fotografija i videa.

Korišćenje `MediaRecorder` API-a

Za snimanje fotografija i videa korišćenjem `MediaRecorder` API-a potrebno je izvršiti sledeće korake:

- Detektovati kameru
- Pristupiti kameri
- Napraviti Preview klasu
- Napraviti Preview raspored
- Podesiti obrađivače događaja
- Snimiti fotografiju ili video u datoteku
- Osloboditi kameru

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" />
9
10   <activity android:name=".CameraActivity" ...>
11     <intent-filter>
12       <action android:name="android.intent.action.MAIN" />
13       <category android:name="android.intent.category.LAUNCHER" />
14     </intent-filter>
15   </activity>
16 </manifest>
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  public static Camera getCameraInstance(){
2    Camera camera = null;
3    try {
4      camera = Camera.open(0); // Attempts to get a Camera instance
5    } catch (Exception e){
6      // Camera is not available (in use or does not exist)
7    }
8
9    return camera; // Returns null if camera is unavailable
10 }
11
```

CameraPreview.java

```
1  public class CameraPreview extends SurfaceView implements SurfaceHolder.
       Callback {
2
3    private SurfaceHolder holder;
4    private Camera camera;
5
6    public CameraPreview(Context context, Camera camera) {
7      super(context);
8      this.camera = camera;
9
10     // Install a SurfaceHolder.Callback so we get notified when the
11     // underlying surface is created and destroyed.
12     holder = getHolder();
13     holder.addCallback(this);
14     // deprecated setting, but required on Android versions prior to 3.0
15     holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
16   }
17
18   public void surfaceCreated(SurfaceHolder holder) {
19     // The Surface has been created, now tell the camera where to draw the
       preview.
20     try {
21       camera.setPreviewDisplay(holder);
22       camera.startPreview();
23     } catch (IOException e) {
24       Log.d(TAG, "Error setting camera preview: " + e.getMessage());
25     }
26   }
27
28   public void surfaceDestroyed(SurfaceHolder holder) {
29     // Take care of releasing the Camera preview in your activity.
30   }
31
```

CameraPreview.java

```
1   public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
2     // If your preview can change or rotate, take care of those events here.
3
4     if (this.holder.getSurface() == null){
5     // Make sure to stop the preview before resizing or reformatting it.
6     // Preview surface does not exist
7       return;
8     }
9
10    // Stop preview before making changes
11    try {
12      camera.stopPreview();
13    } catch (Exception e){
14      // ignore: tried to stop a non-existent preview
15    }
16
17    // Set preview size and make any resize, rotate or
18    // reformatting changes here
19
20    // Start preview with new settings
21    try {
22      camera.setPreviewDisplay(this.holder);
23      camera.startPreview();
24    } catch (Exception e){
25      Log.d(TAG, "Error starting camera preview: " + e.getMessage());
26    }
27  }
28 }
29
```

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    <FrameLayout
9      android:id="@+id/camera_preview"
10     android:layout_width="fill_parent"
11     android:layout_height="fill_parent"
12     android:layout_weight="1" />
13
14   <Button
15     android:id="@+id/button_capture"
16     android:text="Capture"
17     android:layout_width="wrap_content"
18     android:layout_height="wrap_content"
19     android:layout_gravity="center"
20     android:onClick="onClick" />
21
22 </LinearLayout>
23
```

CameraActivity.java

```
1  public class CameraActivity extends Activity {
2
3    private Camera camera;
4    private CameraPreview cameraPreview;
5
6    @Override
7    public void onCreate(Bundle savedInstanceState) {
8      super.onCreate(savedInstanceState);
9      setContentView(R.layout.main);
10
11     // Create an instance of Camera
12     camera = getCameraInstance();
13
14     // Create our Preview view and set it as the content of our activity.
15     cameraPreview = new CameraPreview(this, camera);
16     FrameLayout preview = (FrameLayout) findViewById(R.id.camera_preview);
17     preview.addView(cameraPreview);
18   }
19
20   @Override
21   public void onDestroy() {
22     camera.stopPreview();
23     camera.release();
24   }
25 }
26
```

CameraActivity.java

```
1   private PictureCallback pictureCallback = new PictureCallback() {
2
3     @Override
4     public void onPictureTaken(byte[] data, Camera camera) {
5       File file = getOutputMediaFile(MEDIA_TYPE_IMAGE);
6       if (file == null) {
7         Log.d(TAG, "Error creating media file, check storage permissions: " + e
      .getMessage());
8         return;
9       }
10
11      try {
12        FileOutputStream fos = new FileOutputStream(file);
13        fos.write(data);
14        fos.close();
15      } catch (FileNotFoundException e) {
16        Log.d(TAG, "File not found: " + e.getMessage());
17      } catch (IOException e) {
18        Log.d(TAG, "Error accessing file: " + e.getMessage());
19      }
20    }
21  };
22
23  public void onClick(View v) {
24    // get an image from the camera
25    camera.takePicture(null, null, pictureCallback);
26  }
27
```

CameraActivity.java

```
1   private boolean isRecording = false;
2
3   public void onClick(View v) {
4     if (isRecording) {
5       // Stop recording and release camera
6       mediaRecorder.stop();  // Stops the recording
7       releaseMediaRecorder(); // Releases the MediaRecorder object
8
9       // Inform the user that recording has stopped
10      setCaptureButtonText("Capture");
11      isRecording = false;
12    } else {
13      // Initialize video camera
14      if (prepareVideoRecorder()) {
15        // Camera is available and unlocked, MediaRecorder is prepared,
16        // now you can start recording
17        mediaRecorder.start();
18
19        // inform the user that recording has started
20        setCaptureButtonText("Stop");
21        isRecording = true;
22      } else {
23        // prepare didn't work, release the camera
24        releaseMediaRecorder();
25        // inform user
26      }
27    }
28  }
29
```

CameraActivity.java

```
1   private boolean prepareVideoRecorder(){
2
3     camera = getCameraInstance();
4     mediaRecorder = new MediaRecorder();
5
6     // Step 1: Unlock and set camera to MediaRecorder
7     camera.unlock();
8     mediaRecorder.setCamera(camera);
9
10    // Step 2: Set sources
11    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.CAMCORDER);
12    mediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
13
14    // Step 3: Set a CamcorderProfile (requires API Level 8 or higher)
15    mediaRecorder.setProfile(CamcorderProfile.get(CamcorderProfile.QUALITY_HIGH
      ));
16
17    // Step 4: Set output file
18    mediaRecorder.setOutputFile(getOutputMediaFile(MEDIA_TYPE_VIDEO).toString()
      );
19
20
```

CameraActivity.java

```
1     // Step 5: Set the preview output
2     mediaRecorder.setPreviewDisplay(preview.getHolder().getSurface());
3
4     // Step 6: Prepare configured MediaRecorder
5     try {
6       mediaRecorder.prepare();
7     } catch (IllegalStateException e) {
8       Log.d(TAG, "IllegalStateException preparing MediaRecorder: " + e.
      getMessage());
9       releaseMediaRecorder();
10      return false;
11    } catch (IOException e) {
12      Log.d(TAG, "IOException preparing MediaRecorder: " + e.getMessage());
13      releaseMediaRecorder();
14      return false;
15    }
16    return true;
17  }
18
19  private void releaseMediaRecorder(){
20    if (mediaRecorder != null) {
21      mediaRecorder.reset(); // clear recorder configuration
22      mediaRecorder.release(); // release the recorder object
23      mediaRecorder = null;
24      camera.lock(); // lock camera for later use
25    }
26  }
27
```

Funkcije kamere

- Različite kamere mogu da imaju različite funkcije (autofokus, blic, itd.)
- U toku izvršavanja programa je moguće proveriti da li kamera ima određenu funkciju i podesiti parametre određenih funkcija.

CameraActivity.java

```
1  public void checkCameraFeatures() {
2    // Get Camera parameters
3    Camera.Parameters params = camera.getParameters();
4
5    List<String> focusModes = params.getSupportedFocusModes();
6    if (focusModes.contains(Camera.Parameters.FOCUS_MODE_AUTO)) {
7      // Autofocus mode is supported
8    }
9  }
10
```

CameraActivity.java

```
1 public void setCameraFeatures() {
2   // Get Camera parameters
3   Camera.Parameters params = camera.getParameters();
4   // Set the focus mode
5   params.setFocusMode(Camera.Parameters.FOCUS_MODE_AUTO);
6   // set Camera parameters
7   camera.setParameters(params);
8 }
9
```

Funkcije kamere

Funkcija	Opis
Face Detection	Identifikovanje ljudskih lica unutar fotografije i njihovo korišćenje za automatsko podešavanje fokusa, ekspozicije i "white balance"-a.
Metering Areas	Postavljanje oblasti unutar fotografije koja će se koristiti za podešavanje ekspozicije.
Focus Areas	Postavljanje oblasti unutar fotografije koja će se koristiti za podešavanje fokusa.
White Balance Lock	Pokretanje ili zaustavljanje automatskog podešavanja "white balance"-a.

Table 7: Funkcije kamere.

Funkcije kamere

Funkcija	Opis
Exposure Lock	Pokretanje ili zaustavljanje automatskog podešavanja ekspozicije.
Video Snapshot	Snimanje niza fotografija dok se snima video (frame grab).
Time Lapse Video	Snimanje niza fotografija sa kašnjenjem da bi se snimio "time lapse" video.
Multiple Cameras	Podrška za više od jedne kamere (uključujući prednju i zadnju kameru).
Focus Distance	Određivanje razdaljine između kamere i objekta koji je u fokusu.

Table 8: Funkcije kamere.

Funkcije kamere

Funkcija	Opis
Zoom	Postavljanje nivoa uvećanja snimka.
Exposure Compensation	Povećavanje ili smanjivanje ekspozicije.
GPS Data	Uključivanje ili izostavljanje geografskih koordinata uz fotografiju.
White Balance	Postavljanje "white balance" moda, koji utiče na boju na snimljenoj fotografiji ili snimku.
Focus Mode	Postavljanje fokusa (automatski, fiksni, mikro, beskonačni, itd.).

Table 9: Funkcije kamere.

Funkcije kamere

Funkcija	Opis
Scene Mode	Primena prethodno podešenih parametara za određene tipove scena (kao što su noć, plaža, sneg, itd.)
JPEG Quality	Podesi nivo kompresije za JPEG fotografije.
Flash Mode	Uključi blic, isključi blic ili koristi automatsko podešavanje.
Color Effects	Primeni efekte koji menjaju boju fotografije (crno-bela fotografija, negativ, itd.)
Picture Format	Podesi format fotografije.
Picture Size	Podesi dimenzije fotografije u pikselima.

Table 10: Funkcije kamere.

All images copyrighted by Android Open Source Project (CC BY)