Sprawdzone rozwiązania dla programistów platformy Android!

Android

Receptury

HELION

O'REILLY®

Ian F. Darwin

Tytuł oryginału: Android Cookbook

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-6269-2

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: helion@helion.pl
WWW: http://helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/andrec
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/andrec.zip

Printed in Poland.

Spis treści

ROZDZIAŁ 10.

Multimedia

10.1. Wprowadzenie — multimedia

Ian Darwin

Omówienie

Android to środowisko multimedialne. Standardowo obejmuje odtwarzacze muzyki i filmów, a większość komercyjnych urządzeń obok domyślnych narzędzi obejmuje też ich bardziej wymyślne odpowiedniki, a także odtwarzacze filmów z serwisu YouTube i inne podobne aplikacje. Z receptur z tego rozdziału dowiesz się, jak sterować wybranymi aspektami świata multimediów w Androidzie.

10.2. Odtwarzanie filmów z serwisu YouTube

Marco Dinacci

Problem

Programista chce w urządzeniu umożliwić odtwarzanie filmów z serwisu YouTube.

Rozwiązanie

Na podstawie identyfikatora URI filmu należy utworzyć obiekt `Intent` z akcją `ACTION_VIEW` i uruchomić nową aktywność.

Omówienie

Na listingu 10.1 pokazano kod potrzebny do uruchomienia filmu z serwisu YouTube za pomocą intencji.

Aby kod z tej receptury zadziałał, w urządzeniu użytkownika musi być zainstalowana standardowa aplikacja YouTube.

Listing 10.1. Uruchamianie filmu z serwisu YouTube za pomocą intencji

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    String video_path = "http://www.youtube.com/watch?v=opZ69P-0Jbc";
    Uri uri = Uri.parse(video_path);

    // Ten wiersz powoduje natychmiastowe uruchomienie aplikacji YouTube (jeśli jest
    // zainstalowana). Jeżeli usuniesz ten wiersz, użytkownik zobaczy listę aplikacji
    // do wyboru.
    uri = Uri.parse("vnd.youtube:"  + uri.getQueryParameter("v"));

    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
```

W przykładzie podano standardowy adres URL filmu z serwisu YouTube.com. Człon `uri.getQueryParameter("v")` służy do określania identyfikatora filmu na podstawie identyfikatora URI. W przykładzie identyfikator filmu to `opZ69P-0Jbc`.

10.3. Używanie obiektu Gallery wraz z kontrolką ImageSwitcher

Nidhin Jose Davis

Problem

Programista chce utworzyć interfejs użytkownika do przeglądania kolekcji rysunków.

Rozwiązanie

Pożądany efekt można uzyskać dzięki zastosowaniu obiektu `Gallery` wraz z kontrolką `ImageSwitcher`.

Omówienie

Można wykorzystać obiekt `Gallery` (`android.widget.Gallery`) i kontrolkę `ImageSwitcher` (`android.widget.ImageSwitcher`) do utworzenia eleganckiej przeglądarki rysunków. Na listingu 10.2 pokazano układ dla obiektu `Gallery`.

Listing 10.2. Układ dla obiektu Gallery

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

```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >

        <ImageSwitcher
             android:id="@+id/switcher"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentRight="true"
            android:layout_alignParentBottom="true"
                />

        <Gallery
            android:id="@+id/gallery"
            android:background="#55000000"
            android:layout_width="fill_parent"
            android:layout_height="60dip"
            android:spacing="16px"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true"
            android:gravity="center_vertical"
            />

    </RelativeLayout>
```

Na listingu 10.3 pokazano, jak wykorzystać ten układ.

Listing 10.3. Główna aktywność ImageBrowser z przykładu z galerią

```
    public class ImageBrowser extends Activity
            implements AdapterView.OnItemSelectedListener, ViewSwitcher.ViewFactory {
        private ImageSwitcher mISwitcher;
        private ArrayList<Drawable> allimages = new ArrayList<Drawable>();

        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
                    // Usuwanie paska tytułu
            requestWindowFeature(Window.FEATURE_NO_TITLE);
            setContentView(R.layout.main);

            getImages();

            mISwitcher = (ImageSwitcher)findViewById(R.id.switcher);
            mISwitcher.setFactory(this);
                    // Animacja przy przełączaniu rysunków
            mISwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                    android.R.anim.fade_in));
            mISwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                    android.R.anim.fade_out));

            Gallery gallery = (Gallery) findViewById(R.id.gallery);
            gallery.setAdapter(new ImageAdapter(this));
            gallery.setOnItemSelectedListener(this);
        }

        private void getImages() {
            allimages.add(this.getResources().getDrawable(R.drawable.image1));
            allimages.add(this.getResources().getDrawable(R.drawable.image2));
            allimages.add(this.getResources().getDrawable(R.drawable.image3));
            allimages.add(this.getResources().getDrawable(R.drawable.image4));
            allimages.add(this.getResources().getDrawable(R.drawable.image5));
```

```
        allimages.add(this.getResources().getDrawable(R.drawable.image6));
        allimages.add(this.getResources().getDrawable(R.drawable.image7));
        allimages.add(this.getResources().getDrawable(R.drawable.image8));
        allimages.add(this.getResources().getDrawable(R.drawable.image9));

    }

    @Override
    public void onItemSelected(AdapterView<?> arg0, View v, int position, long id) {
        try{
            mISwitcher.setImageDrawable(allimages.get(position));
        }catch(Exception e){}
    }

    @Override
    public void onNothingSelected(AdapterView<?> arg0) {
        // Pusta
    }

    @Override
    public View makeView() {
          ImageView i = new ImageView(this);
          i.setBackgroundColor(0xFF000000);
          i.setScaleType(ImageView.ScaleType.FIT_CENTER);
          i.setLayoutParams(new ImageSwitcher.LayoutParams(
              ImageSwitcher.LayoutParams.FILL_PARENT,
              ImageSwitcher.LayoutParams.FILL_PARENT));
          return i;
    }

    public class ImageAdapter extends BaseAdapter {
        private Context mContext;

        public ImageAdapter(Context c) {
            mContext = c;
        }

        public int getCount() {
            return allimages.size();
        }

        public Object getItem(int position) {
            return position;
        }

        public long getItemId(int position) {
            return position;
        }

        public View getView(int position, View convertView, ViewGroup parent) {
                    ImageView galleryview = new ImageView(mContext);
                    galleryview.setImageDrawable(allimages.get(position));
                    galleryview.setAdjustViewBounds(true);
                    galleryview.setLayoutParams(new
                        LayoutParams(LayoutParams.WRAP_CONTENT,
                        LayoutParams.WRAP_CONTENT));
                    galleryview.setPadding(5, 0, 5, 0);
                    galleryview.setBackgroundResource(android.R.drawable.picture_frame);
                    return galleryview;
        }
    }
}
```

10.4. Rejestrowanie filmów za pomocą klasy MediaRecorder

Marco Dinacci

Problem

Programista chce za pomocą wbudowanej kamery rejestrować filmy i zapisywać je na dysku.

Rozwiązanie

Należy zarejestrować film i zapisać go w telefonie za pomocą klasy `MediaRecorder` z frameworku Androida.

Omówienie

Do nagrywania dźwięku i filmów służy klasa `MediaRecorder`. Klasa ta ma prosty interfejs API, ponieważ jednak oparta jest na prostej maszynie stanowej, metody trzeba wywoływać w określonej kolejności, tak aby uniknąć wystąpienia wyjątku `IllegalStateException`.

Utwórz nową aktywność i przesłoń metodę `onCreate` za pomocą kodu z listingu 10.4.

Listing 10.4. Metoda onCreate() z głównej aktywności

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.media_recorder_recipe);

    // Film należy nagrywać w orientacji poziomej
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

    mSurfaceView = (SurfaceView) findViewById(R.id.surfaceView);
    mHolder = mSurfaceView.getHolder();
    mHolder.addCallback(this);
    mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);

    mToggleButton = (ToggleButton) findViewById(R.id.toggleRecordingButton);
    mToggleButton.setOnClickListener(new OnClickListener() {
            @Override
            // Rozpoczynanie i wstrzymywanie nagrywania filmu
            public void onClick(View v) {
                    if (((ToggleButton)v).isChecked())
                            mMediaRecorder.start();
                    else {
                            mMediaRecorder.stop();
                            mMediaRecorder.reset();
                            try {
                                    initRecorder(mHolder.getSurface());
                            } catch (IOException e) {
                                    e.printStackTrace();
                            }
                    }
            }
    });
}
```

Klatki z podglądem filmu są wyświetlane w widoku `SurfaceView`. Do sterowania nagrywaniem służy przycisk, który pozwala rozpocząć i wstrzymać rejestrowanie. Po zakończeniu nagrywania należy zatrzymać pracę obiektu `MediaRecorder`. Ponieważ metoda `stop` resetuje stan wszystkich zmiennych maszyny stanowej, więc aby móc rozpocząć rejestrowanie następnego filmu, należy zresetować maszynę stanową i jeszcze raz wywołać metodę `initRecorder`.

W metodzie `initRecorder` aplikacja konfiguruje obiekt `MediaRecorder` oraz aparat, co pokazano na listingu 10.5.

Listing 10.5. Konfigurowanie obiektu MediaRecorder

```
/* Inicjowanie obiektu MediaRecorder. Aby obiekt działał poprawnie, metody
 * trzeba wywoływać w odpowiedniej kolejności
 */
private void initRecorder(Surface surface) throws IOException {
    // Bardzo ważne jest, aby przed wywołaniem metody setCamera odblokować aparat.
    // W przeciwnym razie podgląd będzie niewidoczny
    if(mCamera == null) {
        mCamera = Camera.open();
        mCamera.unlock();
    }

    if(mMediaRecorder == null)
        mMediaRecorder = new MediaRecorder();

    mMediaRecorder.setPreviewDisplay(surface);
    mMediaRecorder.setCamera(mCamera);

    mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
    File file = createFile();

    mMediaRecorder.setOutputFile(file.getAbsolutePath());

    // Bez ograniczeń. Nie zapomnij sprawdzić, ile wolnej pamięci jest na dysku
    mMediaRecorder.setMaxDuration(-1);
    mMediaRecorder.setVideoFrameRate(15);

    mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);

    try {
        mMediaRecorder.prepare();
    } catch (IllegalStateException e) {
        // Zgłaszany, jeśli poprzednie metody nie zostały wywołane w odpowiedniej
        // kolejności
        e.printStackTrace();
    }

    mInitSuccesful = true;
}
```

Ważne jest, aby przed utworzeniem obiektu `MediaRecorder` utworzyć i odblokować obiekt `Camera`. Metody `setPreviewDisplay` i `setCamera` trzeba wywołać natychmiast po utworzeniu obiektu `MediaRecorder`. Koniecznie trzeba określić format i plik wyjściowy. Inne opcje (jeśli występują) należy ustawiać w kolejności przedstawionej na listingu 10.5.

Obiekt `MediaRecorder` najlepiej jest inicjować po utworzeniu powierzchni. Aby otrzymywać powiadomienia o tym, że powierzchnia jest gotowa, aktywność jest rejestrowana jako odbiornik `SurfaceHolder.Callback`. Ponadto w kodzie przesłonięto metodę `surfaceCreated`. Aplikacja wywołuje w niej kod do inicjowania procesu nagrywania.

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    try {
        if(!mInitSuccessful)
            initRecorder(mHolder.getSurface());
        } catch (IOException e) {
            e.printStackTrace();      // Zastosować lepszą obsługę błędów?
        }
}
```

Po zakończeniu korzystania z powierzchni należy pamiętać o zwolnieniu zasobów, ponieważ `Camera` to obiekt współużytkowany i inne aplikacje także mogą z niego korzystać:

```
private void shutdown() {
    // Zwalnianie obiektów MediaRecorder i — przede wszystkim — Camera,
    // ponieważ ten drugi jest współużytkowany i mogą go potrzebować inne programy
    mMediaRecorder.reset();
    mMediaRecorder.release();
    mCamera.release();

    // Po zwolnieniu obiektów nie można ich ponownie wykorzystać
    mMediaRecorder = null;
    mCamera = null;
}
```

Aby przedstawiony wcześniej kod był wywoływany automatycznie po zakończeniu korzystania z aktywności przez użytkownika, należy przesłonić metodę `surfaceDestroyed`:

```
@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    shutdown();
}
```

10.5. Jak wykorzystać androidowy mechanizm wykrywania twarzy?

Wagied Davids

Problem

Programista chce, aby aplikacja wykrywała, czy na danym zdjęciu znajdują się ludzkie twarze, a jeśli tak, to gdzie.

Rozwiązanie

Należy zastosować wbudowany androidowy mechanizm wykrywania twarzy.

Wykrywanie twarzy to atrakcyjna i ciekawa ukryta funkcja interfejsu API Androida. Jest dostępna od Androida 1.5. Wykrywanie twarzy polega na wskazywaniu na zdjęciach fragmentów, które przypominają ludzką twarz. Rozpoznawanie obiektów na podstawie zbioru cech to jedno z zagadnień z obszaru uczenia maszynowego. Warto zauważyć, że nie chodzi tu o funkcję rozpoznawania twarzy. Omawiany tu mechanizm jedynie wykrywa fragmenty wyglądające jak twarz, natomiast nie określa, do kogo ona należy. Dopiero w wersji Ice Cream Sandwich

(Android 4.0) wprowadzono funkcję rozpoznawania twarzy, którą można wykorzystać do odblokowywania telefonu.

Omówienie

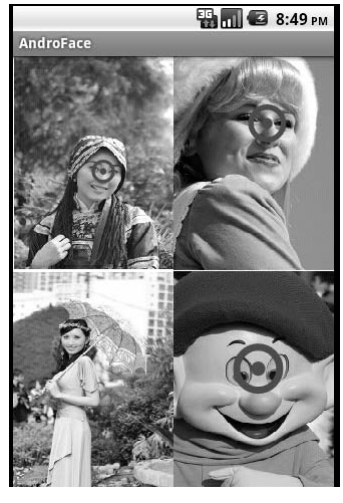
Główna aktywność (przedstawiona na listingu 10.6) tworzy obiekt `FaceDetectionView`. W przykładowej aplikacji sprawdzany plik zapisano na stałe, jednak w produkcyjnej wersji programu zdjęcia powinny pochodzić z aparatu lub galerii.

Listing 10.6. Główna aktywność

```
import android.app.Activity;
import android.os.Bundle;

public class Main extends Activity
{
/** Wywoływana, gdy aktywność tworzona jest po raz pierwszy */
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(new FaceDetectionView(this, "face5.JPG"));
}
}
```

`FaceDetectionView` to niestandardowa klasa, służąca do zarządzania wykrywaniem twarzy z wykorzystaniem klasy `android.media.FaceDetector`. Metoda `init()` określa grafikę używaną do oznaczania twarzy. W przykładzie wiadomo, gdzie znajdują się twarze, a Android powinien je znaleźć. Główne operacje wykonywane są w metodzie `detectFaces()`. Wywołuje ona metodę `findFaces` klasy `FaceDetector`. Do tej ostatniej metody należy przekazać zdjęcie i tablicę na wyniki. Następnie można przejść po znalezionych twarzach. Potrzebny kod pokazano na listingu 10.7, a na rysunku 10.1 widoczny jest efekt działania aplikacji.



Rysunek 10.1. Wykrywanie twarzy w praktyce

Listing 10.7. Plik FaceDetectionView.java

```
...
import android.media.FaceDetector;

public class FaceDetectionView extends View {
    private static final String tag = FaceDetectionView.class.getName();
    private static final int NUM_FACES = 10;
    private FaceDetector arrayFaces;
    private final FaceDetector.Face getAllFaces[] = new FaceDetector.Face[NUM_FACES];
    private FaceDetector.Face getFace = null;

    private final PointF eyesMidPts[] = new PointF[NUM_FACES];
    private final float eyesDistance[] = new float[NUM_FACES];

    private Bitmap sourceImage;

    private final Paint tmpPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    private final Paint pOuterBullsEye = new Paint(Paint.ANTI_ALIAS_FLAG);
    private final Paint pInnerBullsEye = new Paint(Paint.ANTI_ALIAS_FLAG);

    private int picWidth, picHeight;
    private float xRatio, yRatio;
    private ImageLoader mImageLoader = null;

    public FaceDetectionView(Context context, String imagePath) {
        super(context);
        init();
        mImageLoader = ImageLoader.getInstance(context);
        sourceImage = mImageLoader.loadFromFile(imagePath);
        detectFaces();
    }

    private void init() {
        Log.d(tag, "Init()...");
        pInnerBullsEye.setStyle(Paint.Style.FILL);
        pInnerBullsEye.setColor(Color.RED);
        pOuterBullsEye.setStyle(Paint.Style.STROKE);
        pOuterBullsEye.setColor(Color.RED);
        tmpPaint.setStyle(Paint.Style.STROKE);
        tmpPaint.setTextAlign(Paint.Align.CENTER);
        BitmapFactory.Options bfo = new BitmapFactory.Options();
        bfo.inPreferredConfig = Bitmap.Config.RGB_565;
    }

    private void loadImage(String imagePath) {
        sourceImage = mImageLoader.loadFromFile(imagePath);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        Log.d(tag, "onDraw()...");

        xRatio = getWidth() * 1.0f / picWidth;
        yRatio = getHeight() * 1.0f / picHeight;
        canvas.drawBitmap(
            sourceImage, null, new Rect(0, 0, getWidth(), getHeight()), tmpPaint);
        for (int i = 0; i < eyesMidPts.length; i++) {
            if (eyesMidPts[i] != null) {
                pOuterBullsEye.setStrokeWidth(eyesDistance[i] / 6);
                canvas.drawCircle(eyesMidPts[i].x * xRatio,
                    eyesMidPts[i].y * yRatio, eyesDistance[i] / 2, pOuterBullsEye);
```

```
                canvas.drawCircle(eyesMidPts[i].x * xRatio,
                    eyesMidPts[i].y * yRatio, eyesDistance[i] / 6, pInnerBullsEye);
            }
        }
    }

    private void detectFaces() {
        Log.d(tag, "detectFaces()...");

        picWidth = sourceImage.getWidth();
        picHeight = sourceImage.getHeight();

        arrayFaces = new FaceDetector(picWidth, picHeight, NUM_FACES);
        arrayFaces.findFaces(sourceImage, getAllFaces);

        for (int i = 0; i < getAllFaces.length; i++) {
            getFace = getAllFaces[i];
            try {
                PointF eyesMP = new PointF();
                getFace.getMidPoint(eyesMP);
                eyesDistance[i] = getFace.eyesDistance();
                eyesMidPts[i] = eyesMP;

                Log.i("Twarz",
                    i + " " + getFace.confidence() + " " + getFace.eyesDistance() +
                    " " +
                    "Ustawienie: (" + getFace.pose(FaceDetector.Face.EULER_X) + "," +
                    getFace.pose(FaceDetector.Face.EULER_Y) + "," +
                    getFace.pose(FaceDetector.Face.EULER_Z) + ")" +
                    "Punkt na wysokości oczu: (" + eyesMidPts[i].x + "," +
                                                   eyesMidPts[i].y + ")");
            } catch (Exception e) {
                Log.e("Twarz", i + " – brak twarzy");
            }
        }
    }
}
```

10.6. Odtwarzanie muzyki z pliku

Marco Dinacci

Problem

Programista chce odtwarzać pliki dźwiękowe przechowywane w urządzeniu.

Rozwiązanie

Należy utworzyć i odpowiednio skonfigurować obiekty `MediaPlayer` oraz `MediaController`, a następnie podać ścieżkę do pliku. Potem można rozkoszować się muzyką.

Omówienie

Odtwarzanie plików muzycznych jest proste — wystarczy skonfigurować obiekty `MediaPlayer` i `MediaController`.

Najpierw należy utworzyć aktywność z implementacją interfejsu `MediaPlayerControl` (listing 10.8).

Listing 10.8. Początek klasy z implementacją interfejsu MediaPlayerControl

```
public class PlayAudioActivity extends Activity implements MediaPlayerControl {
        private MediaController mMediaController;
        private MediaPlayer mMediaPlayer;
        private Handler mHandler = new Handler();
```

W metodzie `onCreate` trzeba utworzyć i skonfigurować obiekty `MediaPlayer` i `MediaControler`. Pierwszy z tych obiektów wykonuje standardowe operacje na plikach muzycznych — odtwarza je, wstrzymuje i przechodzi do wskazanego miejsca w pliku. Drugi obiekt to widok z przyciskami uruchamiającymi wspomniane operacje za pomocą metod interfejsu `MediaPlayerControl`.

Kod metody `onCreate` przedstawiono na listingu 10.9.

Listing 10.9. Metoda onCreate() odtwarzacza

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mMediaPlayer = new MediaPlayer();
        mMediaController = new MediaController(this);
        mMediaController.setMediaPlayer(PlayAudioActivity.this);
        mMediaController.setAnchorView(findViewById(R.id.audioView));

        String audioFile = "" ;
        try {
            mMediaPlayer.setDataSource(audioFile);
            mMediaPlayer.prepare();
        } catch (IOException e) {
            Log.e("PlayAudioDemo",
                "Nie można odtworzyć pliku " + audioFile + ".", e);
        }

        mMediaPlayer.setOnPreparedListener(new OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mp) {
                    mHandler.post(new Runnable() {
                            public void run() {
                                    mMediaController.show(10000);
                                    mMediaPlayer.start();
                            }
                    });
            }
        });
    }
```

Oprócz skonfigurowania obiektów `MediaController` i `MediaPlayer` w aplikacji trzeba utworzyć anonimowy odbiornik `OnPreparedListener`, aby uruchamiać odtwarzacz tylko wtedy, gdy plik źródłowy jest gotowy do odtworzenia.

Należy też pamiętać o zwolnieniu zasobów obiektu `MediaPlayer` w momencie usuwania aktywności (listing 10.10).

Listing 10.10. Porządkowanie zasobów odtwarzacza

```
    @Override
    protected void onDestroy() {
```

```
        super.onDestroy();
        mMediaPlayer.stop();
        mMediaPlayer.release();
    }
```

Trzeba też zaimplementować interfejs `MediaPlayerControl`. Kod implementacji jest bardzo prosty, co pokazano na listingu 10.11.

Listing 10.11. Implementacja interfejsu MediaPlayerControl

```
    @Override
    public boolean canPause() {
        return true;
    }

    @Override
    public boolean canSeekBackward() {
        return false;
    }

    @Override
    public boolean canSeekForward() {
        return false;
    }

    @Override
    public int getBufferPercentage() {
        return (mMediaPlayer.getCurrentPosition() * 100) / mMediaPlayer.getDuration();
    }

    // Pozostałe metody tylko kierują wywołania do obiektu MediaPlayer
}
```

Na zakończenie należy przesłonić metodę `onTouchEvent`, aby wyświetlać przyciski z obiektu `MediaController` po dotknięciu ekranu przez użytkownika.

Ponieważ obiekt `MediaController` tworzony jest programowo, układ aktywności jest bardzo prosty:

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

10.7. Odtwarzanie dźwięku bez interakcji z użytkownikiem

Ian Darwin

Problem

Programista chce, aby aplikacja odtwarzała pliki dźwiękowe bez interakcji z użytkownikiem.

Rozwiązanie

Aby odtwarzać pliki dźwiękowe bez interakcji z użytkownikiem (bez kontrolek do zmiany głośności, wstrzymywania odtwarzania itd.), wystarczy utworzyć obiekt `MediaPlayer` dla danego pliku i wywołać metodę `start()`.

Omówienie

Jest to najprostsza technika odtwarzania plików dźwiękowych. W porównaniu z aplikacją z receptury 10.6 ten program nie udostępnia użytkownikom kontrolek do sterowania odtwarzaniem. Dlatego zwykle należy udostępnić przynajmniej przycisk *Stop* lub *Anuluj* — zwłaszcza jeśli plik jest długi. Jeżeli jednak w aplikacji chcesz tylko odtwarzać krótkie efekty dźwiękowe, nie musisz udostępniać takich kontrolek.

Dla pliku trzeba utworzyć obiekt `MediaPlayer`. Plik dźwiękowy może znajdować się na karcie SD lub w katalogu *res/raw* aplikacji. Jeśli plik dźwiękowy jest elementem programu, należy umieścić go we wspomnianym katalogu. Tu używany jest plik *res/raw/alarm_sound.3gp*. Referencja do niego to `R.raw.alarm_sound`, a do odtwarzania pliku służy następujący kod:

```
MediaPlayer player = MediaPlayer.create(this, R.raw.alarm_sound);
player.start();
```

Jeśli plik znajduje się na karcie SD, można odtworzyć go w następujący sposób:

```
MediaPlayer player = new MediaPlayer();
player.setDataSource(fileName);
player.prepare();
player.start();
```

Istnieje też metoda pomocnicza, `MediaPlayer.create(Context, URI)`. Metoda ta automatycznie wywołuje metodę `prepare()`.

Aby sterować odtwarzaczem w aplikacji, można wywoływać odpowiednie metody, np. `player.stop()`, `player.pause()` itd. Jeśli chcesz ponownie uruchomić odtwarzacz po jego zatrzymaniu, wywołaj znów metodę `prepare()`.

Do odbierania powiadomień o zakończeniu odtwarzania służy odbiornik `OnCompletionListener`:

```
player.setOnCompletionListener(new OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) {
                Toast.makeText(Main.this,
                        "Zakończono odtwarzanie", Toast.LENGTH_SHORT).show();
        }
});
```

Po ostatecznym zakończeniu korzystania z obiektu `MediaPlayer` należy wywołać metodę `release()` tego obiektu, aby zwolnić pamięć. Jeśli aplikacja tworzy dużo obiektów `MediaPlayer` i nie zwalnia pamięci, może nastąpić wyczerpanie zasobów.

Zobacz także

Aby wykorzystać wszystkie możliwości obiektu `MediaPlayer`, należy poznać jego różne stany i przejścia między nimi. Pomaga to zrozumieć, które metody można wywoływać. Na stronie *http://developer.android.com/reference/android/media/MediaPlayer.html* znajdziesz kompletny diagram stanów obiektu `MediaPlayer`.

10.8. Konwersja mowy na tekst

Corey Sunwold

Problem

Programista chce, aby aplikacja rejestrowała mowę i przetwarzała ją jako tekst.

Rozwiązanie

Jedną z wyjątkowych cech Androida jest wbudowane przetwarzanie mowy na tekst. Zapewnia to alternatywę dla wprowadzania tekstu. Jest to przydatne, ponieważ użytkownicy mają czasem zajęte ręce i nie mogą wpisywać informacji.

Omówienie

Android udostępnia wygodny interfejs API do korzystania z wbudowanej funkcji rozpoznawania mowy. Interfejs ten oparty jest na intencji `RecognizerIntent`.

Przykładowy układ jest bardzo prosty (przedstawiono go na listingu 10.12). W układzie znajduje się tylko kontrolka `TextView` o nazwie `speechText` i kontrolka `Button` o nazwie `getSpeechButton`. Ta ostatnia kontrolka służy do uruchamiania mechanizmu rozpoznawania mowy, a zwracane wyniki pojawiają się w kontrolce `TextView`.

Listing 10.12. Program ilustrujący rozpoznawanie mowy

```
public class Main extends Activity {

    private static final int RECOGNIZER_RESULT = 1234;

    /** Wywoływana, gdy aktywność tworzona jest po raz pierwszy */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button startSpeech = (Button)findViewById(R.id.getSpeechButton);
        startSpeech.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
            Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
            intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
            intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Mowa na tekst");
            startActivityForResult(intent, RECOGNIZER_RESULT);
        }

        });
    }

    /**
     * Obsługa wyników zwróconych przez aktywność przetwarzającą mowę
```

```
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == RECOGNIZER_RESULT && resultCode == RESULT_OK) {
            ArrayList<String> matches = data.getStringArrayListExtra(
                    RecognizerIntent.EXTRA_RESULTS);

            TextView speechText = (TextView)findViewById(R.id.speechText);
            speechText.setText(matches.get(0).toString());
        }

        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Zobacz także

http://developer.android.com/reference/android/speech/RecognizerIntent.html.

10.9. Konwersja tekstu na mowę

Ian Darwin

Problem

Programista chce, aby aplikacja „wypowiadała" napisane słowa, tak aby użytkownik mógł się z nimi zapoznać bez spoglądania na ekran (np. w trakcie prowadzenia samochodu).

Rozwiązanie

Należy zastosować interfejs API TextToSpeech.

Omówienie

Interfejs API TextToSpeech wbudowany jest w Android, choć w niektórych wersjach platformy trzeba doinstalować pliki dźwiękowe.

Aby rozpocząć korzystanie z omawianego mechanizmu, potrzebny jest tylko obiekt `TextToSpeech`. Teoretycznie wystarczy napisać poniższy kod:

```
private TextToSpeech myTTS = new TextToSpeech(this, this);
myTTS.setLanguage(Locale.US);
myTTS.speak(textToBeSpoken, TextToSpeech.QUEUE_FLUSH, null);
myTTS.shutdown();
```

Aby jednak zapewnić poprawne działanie aplikacji, trzeba zastosować kilka intencji. Jedna powinna sprawdzać, czy dane mechanizmu TTS są dostępne (i instalować je, jeśli jest inaczej). Druga jest potrzebna do uruchamiania mechanizmu TTS. Dlatego w praktyce kod powinien wyglądać tak jak na listingu 10.13. Ta ciekawa prosta aplikacja po każdym wciśnięciu przycisku *Mów* wygłasza jedno z kilku banalnych zdań.

Listing 10.13. Program ilustrujący konwersję tekstu na mowę

```
public class Main extends Activity implements OnInitListener {

    private TextToSpeech myTTS;
    private List<String> phrases = new ArrayList<String>();

    public void onCreate(Bundle savedInstanceState) {

        phrases.add("Hello Android, Goodbye iPhone");
        phrases.add("The quick brown fox jumped over the lazy dog");
        phrases.add("What is your mother's maiden name?");
        phrases.add("Etaoin Shrdlu for Prime Minister");
        phrases.add("The letter 'Q' does not appear in
           'antidisestablishmentarianism')");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button startButton = (Button) findViewById(R.id.start_button);
        startButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Intent checkIntent = new Intent();
                checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
                startActivityForResult(checkIntent, 1);
            }
        });
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == 1) {

            if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
                myTTS = new TextToSpeech(this, this); ❶
                myTTS.setLanguage(Locale.US);
            } else {
                // Brak danych mechanizmu TTS. Aplikacja próbuje je zainstalować
                Intent ttsLoadIntent = new Intent();
                ttsLoadIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                startActivity(ttsLoadIntent);
            }
        }
    }

    public void onInit(int status) {
        if (status == TextToSpeech.SUCCESS) {

            int n = (int)(Math.random() * phrases.size());
            myTTS.speak(phrases.get(n), TextToSpeech.QUEUE_FLUSH, null);
        } else if (status == TextToSpeech.ERROR) {
            myTTS.shutdown();
        }
    }
```

❶ Pierwszy argument to kontekst (aktywność), a drugi — odbiornik `OnInitListener`, tu także zaimplementowany w głównej aktywności. Po zainicjowaniu obiektu `TextToSpeech` wywoływany jest wspomniany odbiornik. Metoda `onInit()` tego odbiornika ma powiadamiać o tym, że mechanizm TTS jest gotowy. Przedstawiona tu prosta aplikacja Speaker tylko generuje słowa. W bardziej rozbudowanym programie warto uruchomić wątek lub usługę do obsługi mowy.

Skorowidz

A

B

C

D

E

F

G

I

J

K

L

M

N

O

P

R

S

Ś

T

U

W

X

Y

Z

Programowanie

O'REILLY®

Android. Receptury

Android jest obecnie najpopularniejszą platformą dla telefonów komórkowych i tabletów. Liczba aktywacji to setki tysięcy w ciągu jednego dnia. Skąd wzięła się ta popularność? Nie bez znaczenia jest tu banalnie prosta integracja z serwisami społecznościowymi oraz ogromny wybór aplikacji i sprzętu, w których można przebierać bez końca. Android to również faworyt deweloperów oprogramowania. Dzięki temu, że platforma jest oparta na języku Java, mogą oni błyskawicznie wykorzystać posiadaną wiedzę oraz znane narzędzia. To mieszanka skazana na sukces!

Książka, którą trzymasz w rękach, pomoże Ci odnieść sukces. Należy ona do ulubionej serii programistów — *Receptury*. Znajdziesz tu najlepsze przepisy na rozwiązanie typowych problemów. W trakcie lektury nauczysz się błyskawicznie konfigurować środowisko pracy, tworzyć kopie zapasowe danych aplikacji oraz testować Twoje dzieło. Ponadto sprawdzisz, jak komunikować się między procesami, tworzyć zaawansowaną grafikę oraz wyświetlać materiały multimedialne. Przekonasz się również, jak łatwo uzyskać informacje z odbiornika GPS, sterować diodą LED w urządzeniu oraz przygotować aplikację do dystrybucji i sprzedaży. Książka ta powinna trafić w ręce wszystkich pasjonatów systemu Android!

Sięgnij po tę książkę i:

- sprawdź, jak uzyskać informacje z odbiornika GPS
- błyskawicznie przygotuj swoje środowisko pracy
- twórz zaawansowane elementy graficzne
- przygotuj aplikację do publikacji

Wypróbuj najlepsze przepisy dla Androida!

helion.pl
księgarnia internetowa

Nr katalogowy: 12421

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-246-6269-2

Cena 89,00 zł

9788324662692