ГЛАВА 9

Мультимедиа

Платформа Android — это богатая мультимедийная среда. В стандартную загрузку Android входят музыкальные и видеоплейеры, и большинство коммерческих устройств поставляются с этими или сторонними версиями, а также с устройствами воспроизведения YouTube и др. В рецептах этой главы показано, как управлять некоторыми аспектами мультимедийного мира, который предлагает платформа Android.

9.1. Воспроизведение видео на YouTube

Марко Диначчи

Проблема

Вы хотите воспроизвести видео с сайта YouTube на своем устройстве.

Решение

Учитывая идентификатор URI для воспроизведения видео, создайте с ним намерение `ACTION_VIEW` и запустите новую активность.

Обсуждение

В примере 9.1 показан код, необходимый для запуска видео с сайта YouTube с помощью намерения.

Для того чтобы этот рецепт работал, на устройстве должно быть установлено стандартное приложение YouTube или совместимое с ним.

Пример 9.1. Запуск видео YouTube с намерением

```
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);

    // Благодаря этой инструкции инсталлированное приложение YouTube
    // будет запущено немедленно. Без нее вам будет предложен список
    // приложения для выбора.
    uri = Uri.parse("vnd.youtube:" + uri.getQueryParameter("v"));

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

В этом примере используется стандартный идентификатор URL `YouTube.com`. Для извлечения идентификатора видео из самого идентификатора URI используется метод `Uri.getQueryParameter("v")`. В нашем примере идентификатор — `opZ69P-0Jbc`.

9.2. Захват видео с помощью компонента `MediaRecorder`

Марко Диначчи

Проблема

Вы хотите захватить видео с помощью встроенной камеры устройства и сохранить его на диске.

Решение

Захватите видео и запишите его на телефоне, используя класс `c`, предоставляемый платформой Android.

Обсуждение

Компонент `MediaRecorder` обычно используется для записи аудио и/или видео. Класс имеет простой интерфейс API, но поскольку он основан на простом конечном автомате, методы должны быть вызваны в правильном порядке, чтобы избежать появления ошибок типа `IllegalStateExceptions`.

Создайте новое действие и переопределите метод `onCreate()` с кодом, показанным в примере 9.2.

Пример 9.2. Метод `onCreate()` основной активности

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

// Получаем видео в альбомной ориентации
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
          // Включаем запись видео
          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();
                        }
                  }
          }
     });
}
```

Кадры предварительного просмотра с камеры будут отображаться в компоненте `SurfaceView`. Запись осуществляется с помощью переключателя. По завершении записи мы остановим компонент `MediaRecorder`. Поскольку метод `stop()` сбрасывает все переменные конечного автомата, чтобы иметь возможность захватить другое видео, мы перезапускаем конечный автомат и снова вызываем метод `initRecorder()`. Именно здесь мы настраиваем компонент `MediaRecorder` и камеру, как показано в примере 9.3.

Пример 9.3. Настройка компонента `MediaRecorder`

```
/* Инициализация компонента MediaRecorder. Порядок вызова методов
   очень важен для правильного функционирования.
*/

private void initRecorder(Surface surface) throws IOException {
// Очень важно разблокировать камеру до вызова setCamera(),
// иначе экран предварительного просмотра будет черным

   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());

    // Ограничений нет. Не забудьте проверить пространство на диске.
    mMediaRecorder.setMaxDuration(-1);
    mMediaRecorder.setVideoFrameRate(15);

    mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);

    try {
       mMediaRecorder.prepare();
    } catch (IllegalStateException e) {
         // Это исключение генерируется, если предыдущие вызовы
         // выполнялись в правильном порядке
         e.printStackTrace();
    }

    mInitSuccesful = true;
}
```

Важно создать и разблокировать объект `Camera` до создания компонента `MediaRecorder`. Методы `SetPreviewDisplay()` и `setCamera()` необходимо вызвать сразу после создания компонента `MediaRecorder`. Выбор формата и выходного файла является обязательным. Другие варианты, если они есть, должны быть вызваны в порядке, указанном в примере 9.3.

Компонент MediaRecorder лучше всего инициализировать, когда поверхность уже создана. Мы регистрируем нашу активность как слушатель `SurfaceHolder.Callback`, чтобы получить уведомление об этом и переопределить метод `surfaceCreated()` для вызова нашего кода инициализации.

```
@Override
public void surfaceCreated(SurfaceHolder holder) {
    try {
         if(!mInitSuccessful)
             initRecorder(mHolder.getSurface());
    } catch (IOException e) {
         e.printStackTrace();  // Лучше обработать ошибку?
    }
}
```

Когда вы закончите работу с поверхностью, не забудьте освободить ресурсы, поскольку `Camera` является общим объектом и может использоваться другими приложениями.

```
private void shutdown() {
      // Освобождаем компоненты MediaRecorder и, особенно,
      // которые используются другими приложениями
      mMediaRecorder.reset();
      mMediaRecorder.release();
      mCamera.release();

      // После освобождения объектов их невозможно использовать
      mMediaRecorder = null;
      mCamera = null;
}
```

Переопределите метод `surfaceDestroyed()`, чтобы предыдущий код можно было вызвать автоматически, когда пользователь выполнил операцию.

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

URL-адрес для загрузки исходного кода

Исходный код для этого примера находится в репозитории `https://github.com/IanDarwin/Android-Cookbook-Examples`, в подкаталоге `MediaRecorderDemo` (см. раздел "Получение и использование примеров кода" предисловия).

9.3. Возможности Android для обнаружения лиц

Вагид Дэвидс

Проблема

Вы хотите узнать, содержит ли данное изображение какие-либо лица людей, и если да, то где они находятся.

Решение

Используйте встроенную функцию обнаружения лиц в системе Android.

Обсуждение

Этот рецепт иллюстрирует, как реализовать распознавание лиц в изображениях. Распознавание лиц — прекрасная и забавная скрытая функция API Android. По сути, распознавание лиц является актом распознавания частей изображения, которые кажутся человеческими лицами. Это часть техники машинного обучения распознавания объектов с использованием набора функций.

На самом деле этот механизм не распознает именно лица. Он обнаруживает части изображения, которые являются лицами, но не сообщает, кому принадлежат лица. В версии Android 4.0 и более поздних версиях распознавание лиц используется для разблокировки телефона.

Основная активность (пример 9.4) создает экземпляр нашего компонента `FaceDetectionView`. В этом примере мы скопируем файл, который нужно отсканировать, но в реальной жизни вы, вероятно, захотите захватить изображение с помощью камеры или выбрать изображение из галереи.

Пример 9.4. Основная активность

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

public class Main extends Activity
{
    /** Вызывается при первом создании активности. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new FaceDetectionView(this, "face5.JPG"));
    }
}
```

`FaceDetectionView` — это наш специальный класс, используемый для управления кодом обнаружения лица с помощью класса `android.media.FaceDetector`. Метод `init()` устанавливает некоторые графические изображения, используемые для обозначения лиц, — в этом примере мы знаем, где находятся лица, и надеемся, что система Android их найдет. Реальная работа выполняется в методе `detectFaces()`, где мы вызываем метод `findFaces()` из класса `FaceDetector`, передавая изображение и массив, который будет содержать результаты. Затем мы перебираем найденные лица. В примере 9.5 показан код, а на рис. 9.1 — результат.

Пример 9.5. Файл `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("Face",
                          i + " " + getFace.confidence() + " " +
                          getFace.eyesDistance() + " " +
                          "Pose: (" + getFace.pose(FaceDetector.Face.EULER_X)
                           + "," +
                          getFace.pose(FaceDetector.Face.EULER_Y) + "," +
                          getFace.pose(FaceDetector.Face.EULER_Z) + ")" +
                          "Eyes Midpoint: (" + eyesMidPts[i].x + "," +
                           eyesMidPts[i].y + ")");
                 } catch (Exception e) {
                      Log.e("Face", i + " is null");
                 }
            }
       }
}
```

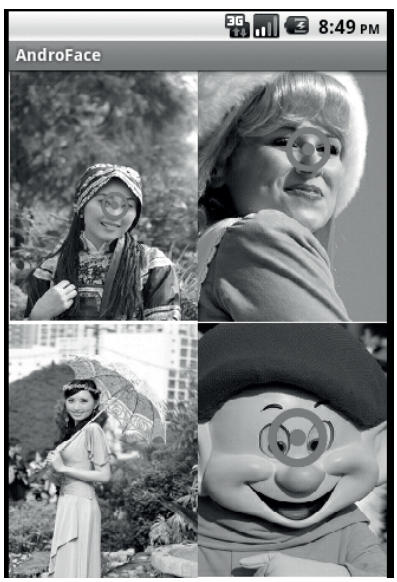

Рис. 9.1. Распознавание лиц в действии

URL-адрес для загрузки исходного кода

Исходный код для этого примера находится в репозитории `https://github.com/IanDarwin/Android-Cookbook-Examples` в подкаталоге `FaceFinder` (см. раздел "Получение и использование примеров кода" предисловия).

9.4. Воспроизведение аудио из файла

Марко Диначчи

Проблема

Вы хотите воспроизвести аудиофайл, хранящийся на устройстве.

Решение

Создайте и правильно настройте компоненты `MediaPlayer` и `MediaController`, укажите путь воспроизведения аудиофайла и наслаждайтесь музыкой.

Обсуждение

Воспроизведение аудиофайла такое же простое, как настройка компонентов `MediaPlayer` и `MediaController`. Сначала создайте новое действие, реализующее интерфейс `MediaPlayerControl` (пример 9.6).

Пример 9.6. Заголовок класса `MediaPlayerControl`

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

В методе `onCreate()` мы создаем и настраиваем компоненты `MediaPlayer` и `MediaController`. Первый — это объект, который выполняет типичные операции с аудиофайлом, такие как воспроизведение, приостановка и поиск. Второй — это представление, содержащее кнопки, запускающие вышеупомянутые операции через наш класс `MediaPlayerControl`. В примере 9.7 показан код `onCreate()`.

Пример 9.7. Метод `onCreate()` класса `MediaPlayer`

```
@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",
         "Could not open file " + audioFile + " for playback.", e);
    }

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

Помимо настройки компонентов `MediaController` и `MediaPlayer`, мы создаем анонимный метод `OnPreparedListener` для запуска проигрывателя только тогда, когда медиа-источник готов к воспроизведению. Не забывайте освобождать компонент `MediaPlayer`, когда активность уничтожена (пример 9.8).

Пример 9.8. Освобождение компонента `MediaPlayer`

```
@Override
protected void onDestroy() {
      super.onDestroy();
      mMediaPlayer.stop();
      mMediaPlayer.release();
}
```

Наконец, мы реализуем интерфейс `MediaPlayerControl`, простой код которого показан в примере 9.9.

Пример 9.9. Реализация интерфейса `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();
}

     // Остальные методы просто делегируются из компонента MediaPlayer

}
```

В качестве последнего штриха переопределяем метод `onTouchEvent()`, чтобы показать кнопки компонента `MediaController`, когда пользователь нажимает на экране. Поскольку мы создаем объект класса `MediaController` программно, компоновка выглядит очень просто:

```
<?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>
```

URL-адрес для загрузки исходного кода

Исходный код для этого примера находится в репозитории `https://github.com/IanDarwin/Android-Cookbook-Examples`, в подкаталоге `MediaPlayerInteractive` (см. раздел "Получение и использование примеров кода" предисловия).

9.5. Воспроизведение аудио без взаимодействия

Ян Дарвин

Проблема

Вы хотите воспроизвести аудиофайл без взаимодействия (например, без настраиваемых пользователем компонентов для регулирования уровня звука, элементов управления паузой/ воспроизведением и т.д.).

Решение

Все, что вам нужно сделать, чтобы воспроизвести файл без взаимодействия, — это создать компонент `MediaPlayer` для файла и вызвать его метод `start()`.

Обсуждение

Это самый простой способ воспроизведения звукового файла. В отличие от рецепта 9.4, эта версия не предлагает пользователю никаких элементов управления для взаимодействия со звуком. Поэтому вы должны предлагать хотя бы кнопку Stop (Стоп) или Cancel (Отмена), особенно если звуковой файл может оказаться длинным. Если вы просто воспроизводите короткий звуковой эффект в своем приложении, такой элемент управления не требуется.

Для вашего файла необходимо создать компонент `MediaPlayer`. Аудиофайл может находиться на SD-карте или в каталоге `res/raw` вашего приложения. Если звуковой файл является частью вашего приложения, сохраните его в каталоге `res/raw`. Предположим, что он находится в каталоге `res/raw/alarm_sound.3gp`; тогда необходимо сослаться на `R.raw.alarm_sound`, и вы можете воспроизвести его следующим образом:

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

Если файл находится на SD-карте, используется следующий вызов:

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

Существует также удобная процедура `MediaPlayer.create(Context, URI)`, которую можно использовать. В любом случае метод `create()` вызовет метод `prepare()` для вас.

Для того чтобы управлять проигрывателем из приложения, можно вызывать соответствующие методы, такие как `player.stop()`, `player.pause()` и т.д. Если вы хотите повторно использовать плейер после его остановки, вы должны снова вызвать метод `prepare()`. Для того чтобы получать уведомление о завершении воспроизведения аудиофайла, используйте метод `OnCompletionListener`:

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

Когда вы работаете с экземпляром класса `MediaPlayer`, вы должны вызывать его метод `release()`, чтобы освободить память. В противном случае, если вы создадите много объектов класса `MediaPlayer`, у вас закончатся ресурсы.

См. также

Для того чтобы эффективно использовать компонент `MediaPlayer`, вы должны понимать его состояния и переходы, так как это поможет вам понять, какие методы

являются подходящими. Документация разработчика (https://developer.android.com/reference/android/media/MediaPlayer.html) содержит полную диаграмму состояний для MediaPlayer.

URL-адрес для загрузки исходного кода

Исходный код для этого примера находится в репозитории https://github.com/IanDarwin/Android-Cookbook-Examples, в подкаталоге MediaPlayerDemo (см. раздел "Получение и использование примеров кода" предисловия).

9.6. Преобразование речи в текст

Кори Сунволл

Проблема

Вы хотите принять речевой ввод и обработать его как текст.

Решение

Одной из уникальных особенностей платформы Android является встроенное преобразование речи в текст. Это обеспечивает альтернативную форму ввода текста для пользователя, который в некоторых ситуациях может не иметь возможности свободно вводить информацию.

Обсуждение

Платформа Android предоставляет простой интерфейс API для использования встроенной функции распознавания голоса с помощью компонента RecognizerIntent. Наш пример компоновки будет очень простым (пример 9.10). Я включил компонент TextView под названием voiceText и кнопку, называемую getSpeechButton. Кнопка будет использоваться для запуска распознавателя голоса, который будет продолжать прослушивание и распознавание, пока пользователь не перестанет говорить в течение нескольких секунд. Когда результаты будут возвращены, они будут отображаться в компоненте TextView.

Пример 9.10. Демонстрационная программа распознавания речи

```
public class Main extends Activity {

    private static final int RECOGNIZER_RESULT = 1234;

    /** Вызывается при первом создании активности. */
    @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, "Speech to text");
                      startActivityForResult(intent, RECOGNIZER_RESULT);
        }

    });
}

    /**
     * Обработка результатов от активности распознавания.
     */

    @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);
    }
}
```

См. также

Документация разработчика по классу `RecognizerIntent` (`https://developer.android.com/reference/android/speech/RecognizerIntent.html`).

URL-адрес для загрузки исходного кода

Исходный код для этого примера находится в репозитории `https://github.com/IanDarwin/Android-Cookbook-Examples`, в подкаталоге `SpeechRecognizerDemo` (см. раздел “Получение и использование примеров кода“ предисловия).

9.7. Воспроизведение голоса устройством после преобразования текста в речь

Ян Дарвин

Проблема

Вы хотите, чтобы ваше приложение произносило слова текста и пользователь мог их воспринимать, не наблюдая за экраном (например, при вождении).

Решение

Используйте интерфейс API TextToSpeech.

Обсуждение

Интерфейс API TextToSpeech (TTS) встроен в систему Android (хотя вам, возможно, придется устанавливать голосовые файлы, в зависимости от используемой версии). Для начала вам нужен объект класса `TextToSpeech`. Теоретически вы могли бы просто сделать это следующим образом:

```
private TextToSpeech myTTS = new TextToSpeech(this, this);
myTTS.setLanguage(Locale.US);

myTTS.speak(textToBeSpoken, TextToSpeech.QUEUE_FLUSH, null);
myTTS.shutdown();
```

Однако для обеспечения успеха вам нужно использовать несколько намерений: одно — для проверки доступности данных TTS, если этот интерфейс инсталлирован, и его инсталляции в противном случае, а другой — для запуска механизма TTS. Таким образом, на практике код должен выглядеть примерно так, как показано в примере 9.11. Это забавное приложение выбирает одну из полдюжины банальных фраз, которые произносятся каждый раз при нажатии кнопки Speak (Говорите).

Пример 9.11. Демонстрационная программа для преобразования текста в речь

```
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 {
              // Данные TTS еще не загружены, инсталлируйте его
              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 Первым аргументом является объект класса `Context` (`Activity`), а вторым — `OnInitListener`, также реализуемый основной активностью в данном случае. Когда инициализация объекта класса `TextToSpeech` завершена, он вызывает слушателя, чей метод `onInit()` предназначен для уведомления о готовности механизма TTS. В этой тривиальной программе Speaker мы просто говорим какую-то фразу. В более длинном примере вы, вероятно, захотите запустить поток или службу для поддержания разговора.

URL-адрес для загрузки исходного кода

Исходный код для этого примера находится в репозитории `https://github.com/IanDarwin/Android-Cookbook-Examples`, в подкаталоге `Speaker` (см. раздел "Получение и использование примеров кода" предисловия).