

*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

© 2013 Helion S.A.

Authorized Polish translation of the English edition of Android Cookbook, 1st Edition, ISBN 9781449388416 © 2012 O'Reilly Media Inc.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Przedmowa	13
1. Podstawowe informacje	19
1.1. Wprowadzenie — podstawowe informacje	19
1.2. Nauka Javy	19
1.3. Tworzenie aplikacji „Witaj, świecie” z poziomu wiersza poleceń	21
1.4. Tworzenie aplikacji „Witaj, świecie” w środowisku Eclipse	24
1.5. Konfigurowanie środowiska IDE w systemie Windows pod kątem programowania aplikacji na Android	29
1.6. Cykl życia w Androidzie	35
1.7. Instalowanie plików .apk w emulatorze za pomocą narzędzia ADB	36
1.8. Instalowanie aplikacji w emulatorze za pomocą sklepu SlideME	38
1.9. Współużytkowanie klas Javy z innym projektem środowiska Eclipse	39
1.10. Wskazywanie bibliotek z implementacją funkcji zewnętrznych	41
1.11. Wykorzystanie przykładów z pakietu SDK do ułatwienia sobie pracy	43
1.12. Aktualizowanie pakietu SDK Androida	46
1.13. Wykonywanie zrzutów w emulatorze i urządzeniu z Androidem	52
1.14. Prosty przykładowy program do odliczania wstecznego	55
1.15. Program Tipster — kalkulator napiwków na Android	57
2. Projektowanie udanych aplikacji	73
2.1. Wprowadzenie — projektowanie udanych aplikacji na Android	73
2.2. Obsługa wyjątków	76
2.3. Obiekt Application w Androidzie jako singleton	79
2.4. Zachowywanie danych po zmianie orientacji ekranu	81
2.5. Monitorowanie poziomu baterii w urządzeniach z Androidem	83
2.6. Tworzenie ekranów powitalnych w Androidzie	84
2.7. Projektowanie aplikacji na potrzeby konferencji, BarCampu, hackathonu lub instytucji	88
2.8. Wykorzystanie narzędzia Google Analytics w aplikacjach na Android	90

2.9. Prosta latarka	92
2.10. Dostosowywanie aplikacji na telefony z Androidem do tabletów	94
2.11. Preferencje obowiązujące przy pierwszym uruchomieniu aplikacji	95
2.12. Formatowanie czasu i daty na potrzeby wyświetlania	97
2.13. Kontrolowanie danych wejściowych za pomocą odbiorników KeyListener	99
2.14. Tworzenie kopii zapasowej danych aplikacji na Android	102
2.15. Stosowanie wskazówek zamiast okien podpowiedzi	108
3. Testy	111
3.1. Wprowadzenie — testy	111
3.2. Programowanie sterowane testami w Androidzie	111
3.3. Konfigurowanie urządzeń AVD na potrzeby testowania aplikacji	112
3.4. Testowanie aplikacji w wielu urządzeniach za pomocą chmury	121
3.5. Tworzenie i stosowanie projektu testowego	122
3.6. Rozwiązywanie problemów z awariami aplikacji	125
3.7. Debugowanie z wykorzystaniem instrukcji Log.d i okna LogCat	128
3.8. Automatyczne otrzymywanie raportów o błędach od użytkowników za pomocą mechanizmu BugSense	129
3.9. Korzystanie z lokalnego dziennika czasu wykonania do analizowania błędów i innych sytuacji	131
3.10. Odtwarzanie scenariuszy cyklu życia aktywności na potrzeby testów	134
3.11. Rozwijanie płynnie działających aplikacji za pomocą narzędzia StrictMode	139
3.12. Korzystanie z programu Monkey	140
3.13. Wysyłanie komunikatów tekstowych i przekazywanie wywołań między urządzeniami AVD	142
4. Komunikacja wewnątrz- i międzyprocesowa	145
4.1. Wprowadzenie — komunikacja wewnątrz- i międzyprocesowa	145
4.2. Obsługiwanie strony internetowej, numeru telefonu lub innych elementów za pomocą intencji	146
4.3. Wysyłanie e-maili z poziomu widoku	147
4.4. Wysyłanie e-maili z załącznikami	150
4.5. Przekazywanie łańcuchów znaków za pomocą instrukcji Intent.putExtra()	151
4.6. Pobieranie danych z aktywności podrzędnej do aktywności głównej	152
4.7. Podtrzymywanie działania usługi w trakcie wyświetlania innych aplikacji	155
4.8. Wysyłanie i odbieranie komunikatów rozgłoszeniowych	157
4.9. Uruchamianie usługi po ponownym uruchomieniu urządzenia	158
4.10. Używanie wątków do tworzenia szybko reagujących aplikacji	159
4.11. Korzystanie z klasy AsyncTask do wykonywania operacji w tle	160
4.12. Przesyłanie komunikatów między wątkami za pomocą kolejki wątków aktywności i komponentu obsługi	166

4.13. Tworzenie androidowej wersji kalendarza Epoch (napisanego w HTML-u i JavaScriptcie)	168
5. Dostawcy treści	175
5.1. Wprowadzenie — dostawcy treści	175
5.2. Pobieranie danych z dostawcy treści	175
5.3. Pisanie dostawcy treści	177
5.4. Pisanie zdalnej usługi na Android	179
6. Grafika	185
6.1. Wprowadzenie — grafika	185
6.2. Stosowanie niestandardowej czcionki	185
6.3. Wyświetlanie obracającego się sześcianu za pomocą specyfikacji OpenGL ES	187
6.4. Sterowanie obracającym się sześcianem	191
6.5. Odręczne rysowanie płynnych linii	194
6.6. Robienie zdjęć za pomocą intencji	198
6.7. Robienie zdjęć za pomocą klasy android.media.Camera	200
6.8. Skanowanie kodu kreskowego lub kodu QR za pomocą programu Google ZXing	204
6.9. Wyświetlanie diagramów i wykresów za pomocą klasy AndroidPlot	207
6.10. Tworzenie ikony do androidowego launchera za pomocą programu Inkscape	208
6.11. Łatwe tworzenie ikon do launchera za pomocą programu Paint.NET i grafik z serwisu OpenClipArt.org	215
6.12. Korzystanie z plików NinePatch	221
6.13. Tworzenie wykresów na strony HTML5 za pomocą biblioteki RGraph	224
6.14. Dodawanie prostej animacji rastrowej	228
6.15. Przybliżanie obrazu za pomocą gestów dotykowych	230
7. Graficzny interfejs użytkownika	235
7.1. Wprowadzenie — interfejs GUI	235
7.2. Poznawanie i przestrzeganie wytycznych tworzenia interfejsu użytkownika	236
7.3. Obsługa zmian konfiguracji przez oddzielenie widoku od modelu	238
7.4. Tworzenie przycisku i odbiornika kliknięć	241
7.5. Pięć sposobów na dołączanie odbiornika zdarzeń	242
7.6. Stosowanie kontrolki CheckBox i RadioButton	246
7.7. Wzbogacanie projektu interfejsu użytkownika za pomocą przycisków graficznych	249
7.8. Udostępnianie listy rozwijanej z opcjami za pomocą klasy Spinner	251
7.9. Obsługa długiego kliknięcia	253
7.10. Wyświetlanie pól tekstowych TextView i EditText	254

7.11. Ograniczanie wartości pola EditText za pomocą atrybutów oraz interfejsu TextWatcher	255
7.12. Kontrolka autoCompleteTextView	257
7.13. Zapełnianie kontrolki autoCompleteTextView za pomocą zapytań do bazy SQLite	259
7.14. Przekształcanie pól tekstowych w pola na hasło	260
7.15. Zmiana klawisza Enter na Next na klawiaturze programowej	261
7.16. Obsługa w aktywności zdarzeń związanych z klawiszami	264
7.17. Pokaż im gwiazdy — kontrolka RatingBar	265
7.18. Drgający widok	268
7.19. Wyświetlanie dotykowych informacji zwrotnych	270
7.20. Przełączanie się między różnymi aktywnościami w widoku TabView	273
7.21. Tworzenie niestandardowego paska tytułu	275
7.22. Formatowanie liczb	277
7.23. Poprawne stosowanie liczby mnogiej	281
7.24. Wyświetlanie drugiego ekranu z poziomu pierwszego	283
7.25. Tworzenie ekranu wczytywania, wyświetlanego przy przełączaniu aktywności	291
7.26. Zakrywanie innych komponentów za pomocą klasy SlidingDrawer	292
7.27. Otwieranie komponentu SlidingDrawer od góry do dołu	295
7.28. Dodawanie do układu obramowania z zaokrąglonymi rogami	296
7.29. Wykrywanie gestów w Androidzie	299
7.30. Tworzenie interfejsu użytkownika w Androidzie 1.6 i nowszych wersjach za pomocą fragmentów z Androida 3.0	305
7.31. Korzystanie z galerii zdjęć w Androidzie 3.0	308
7.32. Tworzenie prostego widżetu aplikacji	311
8. Alerty w interfejsach GUI — menu, okna dialogowe, komunikaty toast i powiadomienia	315
8.1. Wprowadzenie — alerty w interfejsach GUI	315
8.2. Tworzenie i wyświetlanie menu	316
8.3. Obsługa wyboru opcji menu	317
8.4. Tworzenie podmenu	318
8.5. Tworzenie wyskakujących okien dialogowych (okien z alertami)	321
8.6. Kontrolka Timepicker	322
8.7. Tworzenie obrotowego mechanizmu wybierania (podobnego do tego z iPhone'ów)	325
8.8. Tworzenie okna dialogowego z zakładkami	327
8.9. Tworzenie okna ProgressDialog	330
8.10. Tworzenie niestandardowego okna dialogowego z przyciskami, rysunkami i tekstem	331
8.11. Klasa AboutBox do wielokrotnego użytku	333

8.12. Modyfikowanie wyglądu komunikatów toast	336
8.13. Tworzenie powiadomienia wyświetlanego na pasku stanu	337
9. GUI — kontrolka ListView	343
9.1. Wprowadzenie — kontrolka ListView	343
9.2. Używanie kontrolki ListView do tworzenia aplikacji opartych na listach	343
9.3. Tworzenie widoków „brak danych” dla kontrolki ListView	347
9.4. Tworzenie zaawansowanych kontrolki ListView z rysunkami i tekstem	349
9.5. Stosowanie nagłówek sekcji w kontrolkach ListView	352
9.6. Zachowywanie pozycji w kontrolce ListView	356
9.7. Niestandardowy adapter listy	357
9.8. Obsługa zmian orientacji — od wartości z kontrolki ListView po wykresy w orientacji poziomej	360
10. Multimedia	367
10.1. Wprowadzenie — multimedia	367
10.2. Odtwarzanie filmów z serwisu YouTube	367
10.3. Używanie obiektu Gallery wraz z kontrolką ImageSwitcher	368
10.4. Rejestrowanie filmów za pomocą klasy MediaRecorder	371
10.5. Jak wykorzystać androidowy mechanizm wykrywania twarzy?	373
10.6. Odtwarzanie muzyki z pliku	376
10.7. Odtwarzanie dźwięku bez interakcji z użytkownikiem	378
10.8. Konwersja mowy na tekst	380
10.9. Konwersja tekstu na mowę	381
11. Utrwalanie danych	383
11.1. Wprowadzenie — utrwalanie danych	383
11.2. Pobieranie informacji o plikach	383
11.3. Wczytywanie plików z aplikacji, a nie z systemu plików	386
11.4. Wyświetlanie zawartości katalogu	387
11.5. Określanie łącznej ilości pamięci oraz ilości wolnego miejsca na karcie SD	390
11.6. Prosty sposób tworzenia aktywności do ustawiania preferencji użytkownika	390
11.7. Sprawdzanie poprawności ustawień	394
11.8. Zaawansowane wyszukiwanie tekstu	396
11.9. Tworzenie bazy SQLite w aplikacji na Android	401
11.10. Wstawianie danych do bazy SQLite	402
11.11. Wczytywanie wartości z istniejącej bazy SQLite	402
11.12. Praca z datami w bazie SQLite	403
11.13. Przetwarzanie danych w formacie JSON za pomocą klasy JSONObject	406
11.14. Przetwarzanie dokumentów XML za pomocą interfejsu DOM API	407
11.15. Przetwarzanie dokumentów w formacie XML z wykorzystaniem interfejsu XmlPullParser	409

11.16. Dodawanie danych kontaktowych	412
11.17. Wczytywanie danych kontaktowych	415
12. Aplikacje do obsługi połączeń telefonicznych	417
12.1. Wprowadzenie — aplikacje do obsługi połączeń telefonicznych	417
12.2. Wykonywanie operacji w momencie, gdy dzwoni telefon	418
12.3. Przetwarzanie wychodzących połączeń telefonicznych	421
12.4. Wybieranie numeru telefonu	424
12.5. Wysyłanie jedno- lub wieloczęściowych wiadomości SMS	425
12.6. Odbieranie wiadomości SMS w aplikacjach na Android	428
12.7. Wysyłanie wiadomości SMS do emulatora za pomocą okna Emulator Control	429
12.8. Korzystanie z androidowej klasy TelephonyManager do pobierania informacji o urządzeniu	430
13. Aplikacje sieciowe	441
13.1. Wprowadzenie — sieć	441
13.2. Stosowanie usług sieciowych typu RESTful	442
13.3. Używanie wyrażeń regularnych do wyodrębniania informacji z nieustrukturyzowanego tekstu	444
13.4. Przetwarzanie danych z kanałów RSS i Atom za pomocą parsera ROME	446
13.5. Korzystanie ze skrótów MD5 do przetwarzania zwykłego tekstu	450
13.6. Przekształcanie tekstu na odnośniki	451
13.7. Dostęp do stron internetowych za pomocą kontrolki WebView	452
13.8. Modyfikowanie wyglądu kontrolki WebView	453
14. Gry i animacje	455
14.1. Wprowadzenie — gry i animacje	455
14.2. Tworzenie gier na Android za pomocą frameworku flixel-android	456
14.3. Tworzenie gry na Android za pomocą narzędzia AndEngine (Android-Engine)	458
14.4. Przetwarzanie danych wejściowych wprowadzonych w określonym czasie	464
15. Sieci społecznościowe	467
15.1. Wprowadzenie — sieci społecznościowe	467
15.2. Integrowanie aplikacji z sieciami społecznościowymi za pomocą protokołu HTTP	467
15.3. Wczytywanie chronologicznych list tweetów za pomocą formatu JSON	470
16. Lokalizacja i mapy	473
16.1. Wprowadzenie — aplikacje wykorzystujące lokalizację	473
16.2. Pobieranie danych o lokalizacji	473
16.3. Dostęp do danych z GPS-a w aplikacjach	475

16.4. Podawanie fikcyjnych współrzędnych GPS w urządzeniu	477
16.5. Geokodowanie i geokodowanie odwrotne	479
16.6. Przygotowania do korzystania z map Google'a	480
16.7. Wyświetlanie aktualnej lokalizacji urządzenia na mapach Google'a	486
16.8. Wyświetlanie znacznika lokalizacji w widoku MapView	487
16.9. Wyświetlanie kilku znaczników w widoku MapView	490
16.10. Tworzenie warstw dla widoku MapView	495
16.11. Zmienianie trybów widoku MapView	496
16.12. Wyświetlanie ikony na warstwie bez korzystania z obiektów Drawable	497
16.13. Implementowanie wyszukiwania lokalizacji na mapach Google'a	501
16.14. Wyświetlanie widoku MapView w kontrolce TabView	503
16.15. Obsługa długich kliknięć w widokach MapView	505
16.16. Korzystanie z map OpenStreetMap	509
16.17. Tworzenie warstw dla map OSM	511
16.18. Stosowanie skali w mapach OSM	513
16.19. Obsługa dotknięć warstwy mapy OSM	514
16.20. Aktualizowanie lokalizacji na mapach OSM	516
17. Akcelerometr	521
17.1. Wprowadzenie — czujniki	521
17.2. Wykrywanie obecności lub braku czujnika	521
17.3. Wykorzystywanie akcelerometru do wykrywania potrząsania urządzeniem	522
17.4. Używanie akcelerometru do sprawdzania, czy ekran skierowany jest w dół, czy w górę	526
17.5. Określanie ułożenia telefonu z Androidem za pomocą czujnika orientacji	527
17.6. Odczyt wskazań czujnika temperatury	528
18. Bluetooth	531
18.1. Wprowadzenie — Bluetooth	531
18.2. Włączanie Bluetootha i umożliwianie wykrywania urządzenia	532
18.3. Podłączanie urządzenia z Bluetoothem	533
18.4. Oczekiwanie na żądania połączenia Bluetooth oraz ich akceptowanie	536
18.5. Implementowanie wykrywania urządzeń z Bluetoothem	537
19. Sterowanie systemem i urządzeniem	539
19.1. Wprowadzenie — sterowanie systemem i urządzeniem	539
19.2. Dostęp do informacji o sieci i połączeniu	539
19.3. Pobieranie informacji z pliku manifestu	540
19.4. Zmienianie trybu dzwonka telefonu na cichy, wibracje lub normalny	541
19.5. Kopiowanie tekstu i pobieranie go ze schowka	542
19.6. Powiadomienia oparte na diodach LED	544

19.7. Włączanie wibracji w urządzeniu	544
19.8. Uruchamianie poleceń powłoki z poziomu aplikacji	545
19.9. Określanie, czy dana aplikacja jest uruchomiona	546
20. Inne języki programowania i frameworki	549
20.1. Wprowadzenie — inne języki programowania	549
20.2. Uruchamianie zewnętrznych, natywnych instrukcji Uniksa i Linuksa	550
20.3. Uruchamianie kodu natywnego w języku C lub C++ za pomocą interfejsu JNI z pakietu NDK	551
20.4. Wprowadzenie do aplikacji Scripting Layer for Android (SL4A; wcześniej Android Scripting Environment)	556
20.5. Tworzenie alertów za pomocą biblioteki SL4A	558
20.6. Pobieranie dokumentów Google i wyświetlanie ich w kontrolce ListView za pomocą biblioteki SL4A	562
20.7. Używanie kodów QR do rozpowszechniania skryptów SL4A	563
20.8. Używanie JavaScriptu do wykorzystania wbudowanych funkcji telefonu poprzez kontrolkę WebView	566
20.9. Tworzenie aplikacji niezależnych od platformy za pomocą frameworku PhoneGap	568
21. Łańcuchy znaków i internacjonalizacja	571
21.1. Wprowadzenie — internacjonalizacja	571
21.2. Internacjonalizacja tekstu aplikacji	572
21.3. Wyszukiwanie i tłumaczenie łańcuchów znaków	575
21.4. Nuanse związane z plikami strings.xml	577
22. Tworzenie pakietów, instalowanie, dystrybucja i sprzedaż aplikacji	583
22.1. Wprowadzenie — tworzenie pakietów, instalowanie i dystrybucja	583
22.2. Tworzenie certyfikatu używanego przy podpisywaniu	583
22.3. Podpisywanie aplikacji	586
22.4. Udostępnianie aplikacji w sklepie Google Play (dawny Android Market)	587
22.5. Integrowanie sieci AdMob z aplikacją	588
22.6. Zaciemnianie i optymalizowanie kodu za pomocą ProGuarda	592
22.7. Odnośniki do aplikacji ze sklepu Google Play	595
Skorowidz	599

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 multimedialnego 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.OnItemClickListener, 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.setOnItemClickListener(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
        // Rozpocznianie 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();
    }

    mInitSuccessful = 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(!InitSuccessful)
            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

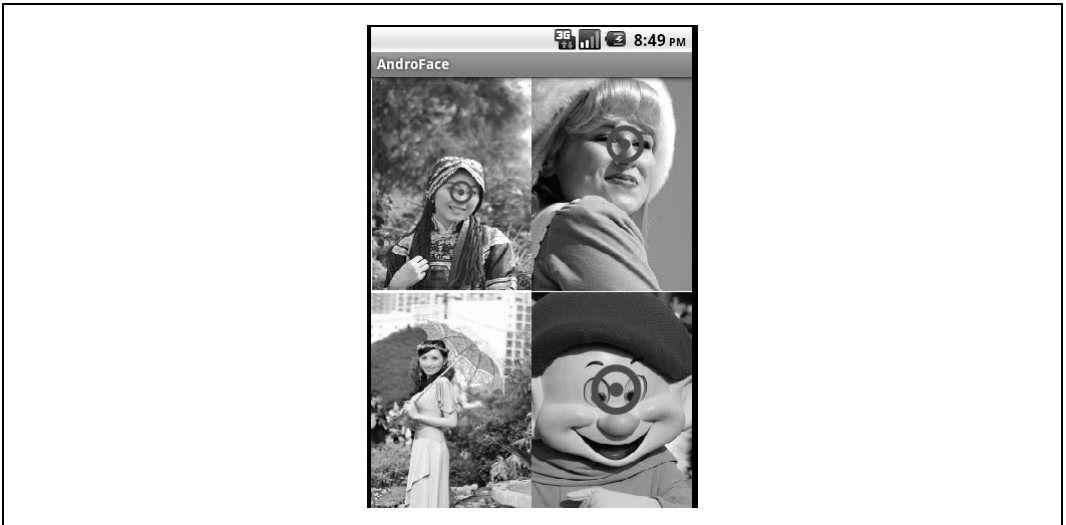
Główna aktywność (przedstawiona na listingu 10.6) tworzy obiekt `FaceDetectionView`. W przykładowej aplikacji sprawdzany plik zapisano na stałe, jednak w produkcyjnej wersji programu zdjęcia powinny pochodzić z aparatu lub galerii.

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 `MediaController`. 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();
        mediaPlayer.stop();
        mediaPlayer.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 kontrolki 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 kontrolki 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 kontrolki.

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.

A

- ADB, 37
- AdMob, 589
- akcelerometr, 521
 - czujniki, 521
 - dostępność czujnika, 522
 - executeShakeAction(), 525
 - getSensorList(), 522
 - isAccelerationChanged(), 525
 - onAccuracyChanged(), 527
 - onSensorChanged(), 523, 526, 527
 - orientacji, 527
 - SensorManager, 522
 - temperatury, 528
 - wykorzystywanie danych, 524
- alerty, 315
 - komunikaty toast, 336
 - powiadomienia, 337
 - dźwięk, 339
 - reakcja na kliknięcie, 342
 - powiadomienie, 341
 - włączanie diody LED, 340
 - wyświetlanie, 338
 - przetwarzanie w tle, 330
 - Python, 558
 - Service, 338
 - wyskakujące okna dialogowe, 321
 - AlertDialog, 321, 532
- AndEngine, 458
- Android, 13
 - akcelerometr, 521
 - czujniki, 521
 - animacje, 455
 - aplikacje, 35
 - awarie, 125
 - Backup Manager, 102
 - cykl życia, 35
 - dostosowywanie do tabletów, 94
 - formatowanie czasu i daty, 97
 - Google Analytics, 90
 - kontrolowanie danych wejściowych, 99
 - latarka, 92
 - lokalizacja, 473
 - mapy, 473
 - menu kontekstowe, 75
 - menu opcji, 75
 - monitorowanie poziomu baterii, 83
 - możliwe stany, 35
 - obsługa połączeń telefonicznych, 417
 - obsługa wyjątków, 76
 - projektowanie, 73
 - sieciowe, 441
 - singleton, 79
 - stoper z odliczaniem wstecznym, 55
 - stosowanie wskazówek, 108
 - Tipster, 57
 - tworzenie ekranów powitalnych, 84
 - tworzenie kopii zapasowej, 102
 - współużytkowanie danych, 80
 - wymagania, 73
 - zmiana orientacji ekranu, 81
 - zrzutu ekranu, 54
- biblioteki, 41
 - AndroidPlot, 41, 207
 - HttpClient, 442, 443
 - OpenStreetMap, 41
 - RGraph, 224
- Bluetooth, 531
 - podłączanie innego urządzenia, 533
 - włączanie, 532
 - wykrywanie urządzeń, 537
 - żądania połączeń, 536
- cykl życia, 35
 - diagram stanów, 36
- ekran powitalny, 86
- formatowanie liczb, 277
 - Double.toString(), 277
 - kody formatujące, 278

- Android
 - formatowanie liczb
 - liczba mnoga, 281
 - metody formatujące, 278
 - fragment, 305
 - gęstość ekranu, 74
 - grafika, 185
 - animacja rastrowa, 228
 - animacje, 455
 - niestandardowa czcionka, 185, 188
 - OpenGL ES, 185, 188
 - paleta kolorów, 216
 - przybliżanie obrazu, 230
 - rysowanie płynnych linii, 194
 - skanowanie kodów, 204
 - tło domyślnego widoku, 221
 - trójwymiarowa, 188
 - tworzenie ikony, 208, 215
 - tworzenie wykresów, 224
 - wyświetlanie danych, 207
 - zdjęcia, 198, 200
 - gry, 455
 - frameworki, 455
 - GUI, 58, 235
 - alerty, 315
 - animacja, 269
 - czas, 322
 - ekran wczytywania, 291
 - komponenty, 58
 - kontrolka ListView, 343
 - lista rozwijana, 251
 - menu, 316
 - niestandardowe okno dialogowe, 328
 - obrotowy mechanizm wybierania, 325
 - obsługa długiego kliknięcia, 253
 - obsługa zmian konfiguracji, 238
 - odbiornik zdarzeń, 242
 - otwieranie nowego ekranu, 283
 - pole wyboru, 247
 - przekształcanie pól tekstowych, 260
 - przycisk wysyłania, 262
 - przyciski graficzne, 249
 - przyciski opcji, 247
 - tworzenie przycisku, 241
 - widżet, 236
 - widżet aplikacji, 236
 - wykrywanie gestów, 299
 - wyświetlanie pól tekstowych, 254
 - zakrywanie innych komponentów, 292
 - internacjonalizacja, 571
 - łańcuchy znaków, 575
 - tekst aplikacji, 572
 - Java, 19
 - pakiet JDK, 29
 - pomijane interfejsy API, 20
 - RGraph, 224
 - środowisko IDE Eclipse, 29
 - kanały danych, 76
 - komunikacja, 145
 - AsyncTask, 145
 - dostawcy treści, 175, 178
 - e-mail, 147
 - intencje, 145
 - IPC, 179
 - komponenty obsługi, 145
 - komunikaty ogłoszeniowe, 157
 - odbiorniki ogłoszeniowe, 145
 - podtrzymywanie działania usługi, 155
 - przekazywanie łańcuchów znaków, 151
 - używanie wątków, 159
 - kontrolki, 59
 - rozmieszczanie, 59
 - mechanizmy wprowadzania danych, 75
 - multimedia, 367
 - Gallery, 368
 - mechanizm wykrywania twarzy, 373
 - odtwarzanie plików muzycznych, 377
 - rejestrowanie filmów, 371
 - YouTube, 367
 - NinePatch, 221
 - pakiet ADK, 21
 - pakiet SDK, 24
 - aktualizacja, 46
 - przykładowe programy, 44
 - wtyczka ADT, 29
 - sterowanie, 539
 - kopiowanie tekstu, 542
 - menedżer aktywności, 547
 - polecenia powłoki, 545
 - połączenie z siecią, 539
 - tryb dzwonka, 541
 - ustawienia projektu, 540
 - włączanie wibracji, 544
 - wyświetlanie powiadomień, 544
 - środowisko Eclipse, 24
 - parametry nowego projektu, 25
 - tworzenie nowego projektu, 45
 - uruchamianie projektu, 27
 - testy, 111
 - awarie, 125
 - BugSense, 129
 - chmura, 121
 - cykl życia, 134
 - debugowanie kodu, 128
 - konfigurowanie urządzeń AVD, 112
 - lokalny dziennik czasu wykonania, 131
 - projekt testowy, 122
 - sterowanie programowaniem, 111

- urządzenia AVD, 23
 - konfigurowanie, 112
- utrwalanie danych, 383
 - baza SQLite, 401, 402, 403
 - dodawanie danych kontaktowych, 412
 - pobieranie informacji o plikach, 383
 - ustawianie preferencji, 390
 - wczytywanie danych kontaktowych, 415
 - wczytywanie plików z aplikacji, 386
 - wyszukiwanie tekstu, 396
 - wyświetlanie zawartości katalogu, 387
- wielkość ekranu, 74
- Windows, 29
 - środowisko IDE Eclipse, 29
- wtyczka ADT, 24
- wyświetlanie czasu i daty, 97
 - DateFormat, 97
 - DateUtils, 99
 - Formatter, 99
 - java.util.Date, 98
 - TextView, 97
 - Time, 99
- zachowywanie danych, 81
 - getLastNonConfigurationInstance(), 81
 - onCreate(), 82
 - onRetainNonConfigurationInstance(), 81
 - onSaveInstanceState(), 81
- zdalna usługa, 179
 - bindService(), 181
 - invokeService(), 182
 - onBind(), 179
 - onCreate(), 180
 - onDestroy(), 180
 - releaseService(), 182
 - startService(), 181
 - stopService(), 182
- zrzut ekranu, 52
 - Device Screen Capture, 53
- Android Compatibility, 305
- Android Localizer, 575
- aplikacje, 35
 - akcelerometr
 - czujnik orientacji, 527
 - dostępność, 522
 - pobieranie danych, 523
 - wykorzystywanie danych, 524
- awarie, 125
 - adb logcat, 125
 - NPE, 126
- Backup Manager, 102
 - implementacja, 102
 - polecenie bmgr, 107
- Bluetooth
 - akceptowanie połączeń, 536
 - pobieranie adresu MAC, 532
 - pobieranie nazwy urządzenia, 532
 - tworzenie serwera, 536
 - wymiana danych z urządzeniem, 534
- certyfi­kat, 583
 - generowanie, 584
 - korzyści, 584
- cykl życia
 - rejestrowanie zdarzeń, 135
- dostawca treści, 178
- dostosowywanie do tabletów, 94
- Google Analytics, 90
- Java
 - kod natywny, 552
- język C
 - kod natywny, 553
- kontrolka ListView, 343
 - nagłówki sekcji, 352
 - obsługa zmian orientacji, 360
 - wyświetlanie listy wierszy, 344
 - zachowywanie pozycji, 356
- kontrolowanie danych wejściowych, 99
 - KeyListener, 99
 - typy odbiorników, 102
- konwersja tekstu na mowę, 382
- kopia zapasowa, 102
 - Backup Manager, 102
- latarka, 92
 - etapy rozwijania, 92
- lokalizacja, 473
 - aktualizowanie lokalizacji, 474
 - fikcyjne współrzędne GPS, 477
 - geokodowanie, 479
 - geokodowanie odwrotne, 480
 - OpenStreetMap, 473
 - pobieranie danych o lokalizacji, 474
 - podawanie fikcyjnej lokalizacji, 478
 - urządzenia GPS, 473
- mapy, 473
 - mapy Google'a, 480
 - AddressOverlay, 495
 - ItemizedOverlay, 491
 - MapTest, 481
 - MyOverlay, 488
 - obsługa długich kliknięć, 506
 - wyszukiwanie lokalizacji, 502
 - wyświetlanie ikony, 499
 - wyświetlanie warstwy, 495
- menu kontekstowe, 75
- menu opcji, 75

- aplikacje
 - obsługa połączeń telefonicznych, 417
 - pobieranie danych, 430
 - połączenia przychodzące, 418, 420
 - połączenia wychodzące, 421
 - wiadomości SMS, 425
 - wybieranie numeru telefonu, 424
 - odbiornik dotyku, 231
 - odnośniki, 595
 - OpenStreetMap, 509
 - aktualizowanie lokalizacji, 517
 - obsługa dotknięć warstwy, 515
 - optymalizowanie kodu, 593
 - pobieranie dokumentów Google, 562
 - podpisywanie, 586
 - projektowanie, 73
 - ekrany powitalne, 84
 - formatowanie czasu i daty, 97
 - funkcje urządzenia, 75
 - kanały danych, 76
 - kontrolowanie danych wejściowych, 99
 - latarka, 92
 - mechanizmy wprowadzania danych, 75
 - obsługa wyjątków, 76
 - odbiornik rozgłoszeniowy, 83
 - wielkość i gęstość ekranu, 74
 - przekształcanie wyjątków, 78
 - rozpoznawanie mowy, 380
 - sieciowe, 441
 - parser danych, 446
 - przekształcanie tekstu, 450, 451
 - RESTful, 442
 - wyrażenia regularne, 444
 - wyświetlanie stron internetowych, 452
 - skanowanie kodów, 204
 - SCAN_FORMATS, 206
 - SCAN_MODE, 206
 - SL4A, 556
 - śledzenie korzystania, 90, 91
 - Google Analytics, 90
 - udostępnianie, 587
 - przesyłanie, 587
 - rejestrwanie, 587
 - współużytkowanie danych, 80
 - wykrywanie gestów, 299
 - wymagania, 73
 - wyświetlanie danych, 207
 - wyświetlanie reklam, 588
 - zaciemnianie kodu, 593
 - aplikacje sieciowe, 441
 - parser danych, 446
 - getRSS(), 447
 - przekształcanie tekstu, 450
 - kontrolka TextView, 451
 - MD5, 450
 - na odnośniki, 451
 - na postać nieczytelną, 450
 - stosowanie RESTful, 442
 - converse(), 442
 - HTTP GET, 442
 - HTTP POST, 442
 - klient usługi, 443
 - URL, 442
 - URLConnection, 442
 - wyrażenia regularne, 444
 - BookRank, 445
 - wyświetlanie stron internetowych, 452
 - kontrolka WebView, 452
- ## B
- Bluetooth, 531
 - podłączanie innego urządzenia, 533
 - wymiana danych z urządzeniem, 534
 - włączanie, 532
 - AlertDialog, 532
 - isEnabled(), 532
 - konfigurowanie, 533
 - onActivityResult(), 532
 - pobieranie adresu MAC, 532
 - pobieranie nazwy, 532
 - wykrywanie urządzeń, 537
 - tryb parowania, 537
 - żądania połączeń, 536
 - akceptowanie połączeń, 536
 - listenUsingRfcommWithServiceRecord(), 536
 - start(), 536
 - tworzenie serwera, 536
 - BugSense, 129
- ## C
- cykl życia, 35
 - android.Activity, 35
 - diagram stanów, 36
 - rejestrwanie zdarzeń, 135
- ## D
- dane, 383
 - baza SQLite, 401
 - data i czas, 403
 - insert(), 402
 - moveToFirst(), 403
 - moveToNext(), 403

- dane
 - baza SQLite
 - onCreate(), 401
 - pobieranie danych, 402
 - query(), 403
 - SQLiteOpenHelper, 401
 - strftime(), 403
 - tworzenie, 401
 - zapisywanie danych, 402
 - informacje o plikach, 383
 - File, 384
 - metody zwracające informacje, 384
 - JSON, 406
 - generowanie danych, 406
 - przetwarzanie łańcucha znaków, 407
 - kontaktowe, 412
 - addContact(), 413
 - dodawanie danych, 412
 - pobieranie danych, 415
 - parser danych, 446
 - pobieranie, 430
 - akcelerometr, 523
 - baza SQLite, 402
 - dane kontaktowe, 415,
 - dostawcy treści, 175
 - lokalizacja, 574
 - połączenia telefoniczne, 430
 - TelephonyManager, 430
 - pojemność karty SD, 390
 - ustawianie preferencji, 390
 - domyślne, 394
 - getBoolean(), 393
 - getDefaultSharedPreferences(), 393
 - getString(), 393
 - onCreate(), 392
 - onSharedPreferenceChanged(), 394
 - PreferenceActivity, 394
 - PreferenceCategory, 391
 - PreferenceScreen, 391, 392
 - wczytywanie plików z aplikacji, 386
 - openRawResource(), 387
 - wyszukiwanie tekstu, 396
 - DbAdapter, 396
 - wyświetlanie zawartości katalogu, 387
 - accept(), 389
 - FilenameFilter, 389
 - kontrolka ListView, 388
 - listFiles(), 388
 - XML, 407
 - interfejs DOM API, 407
 - interfejs XmlPullParser, 409
 - newInstance(), 410
 - newPullParser(), 410
 - przetwarzanie kodu, 408
 - require(), 411
 - statyczne zasoby, 411
 - dokumenty Google, 562
 - lista dokumentów, 564
 - pobieranie, 562
 - dostawcy treści, 175, 178
 - MyContentProvider, 177
 - pobieranie danych, 175
 - getContentResolver(), 177
 - onActivityResult(), 176
 - query(), 177
 - D-pad, 192
 - Droid, 185, 186
- E**
- EDGE, 441
- F**
- Facebook, 467
 - fragment, 305
 - frameworki, 455, 549
 - AndEngine, 458
 - AppCelerator Titanium, 549
 - Flixel, 456
 - PhoneGap, 549, 568
- G**
- Gallery, 368
 - geokodowanie, 479
 - gesty dotykowe, 230
 - odbiornik dotyku, 231
 - Google Analytics, 90
 - Google Play, 587
 - GPRS, 441
 - graficzny interfejs użytkownika, *Patrz* GUI
 - grafika, 185
 - animacja rastrowa, 228
 - onWindowFocusChanged(), 229
 - start(), 229
 - niestandardowa czcionka, 185, 188
 - Iceberg, 186
 - OTF, 186
 - TTF, 186
 - Typeface.create(), 186
 - ustawianie, 187
 - OpenGL ES, 185
 - przybliżanie obrazu, 230
 - odbiornik dotyku, 231

- grafika
 - rysowanie płynnych linii, 194
 - expandDirtyRect(), 195
 - getHistoricalX(int), 195
 - getHistoricalY(int), 195
 - getHistorySize(), 194
 - invalidate(), 195
 - TouchEvent, 194
 - skanowanie kodów, 204
 - SCAN_FORMATS, 206
 - SCAN_MODE, 206
 - tło domyślnego widoku, 221
 - kontrolka EditText, 222
 - trójwymiarowa, 188
 - buffer.position(0), 191
 - D-pad, 192
 - obracanie sześcianu, 192
 - onDrawFrame(), 190
 - onSurfaceChanged(), 190
 - requestFocus(), 194
 - setFocusableInTouchMode(), 194
 - wyświetlanie sześcianu, 190
 - tworzenie ikony, 208, 215
 - formaty ikon, 220
 - obramowanie, 217
 - paleta kolorów, 216
 - PNG, 209
 - SVG, 210
 - wielkość, 210
 - wymiary grafiki, 217
 - wymiary ikon, 217
 - tworzenie wykresów, 224
 - RGraph, 224
 - wyświetlanie danych, 207
 - AndroidPlot, 207
 - zdjęcia, 198, 200
 - android.media.Camera, 200
 - configure(), 202
 - onActivityResult(), 199
 - surfaceChanged(), 202
- gry, 455
 - frameworki, 455
 - AndEngine, 458
 - Flixel, 456
- GUI, 235, 238
 - alerty, 315
 - AlertDialog, 321
 - komunikaty toast, 336
 - powiadomienia, 337
 - przetwarzanie w tle, 330
 - wyskakujące okna dialogowe, 321
 - animacja, 269
 - getCurrentFocus(), 269
 - kod animacji, 269
 - onClick(), 269
 - autouzupełnianie tekstu, 257, 258
 - AutoCompleteTextView, 257
 - onTextChanged(), 258
 - czas, 322
 - kontrolka Timepicker, 323
 - ustawianie czasu, 324
 - dołączanie odbiornika zdarzeń, 242
 - anonimowa klasa wewnętrzna, 244
 - implementacja aktywności, 244
 - interfejs jako typ, 243
 - klasa składowa, 243
 - klasa wewnętrzna, 242
 - new OnClickListener(){...}, 243
 - dotykowe informacje zwrotne, 270
 - oparte na wibracjach, 271
 - zdarzenia generujące, 272
 - działanie widoków, 249
 - ekran wczytywania, 291
 - LoadingScreen, 292
 - fragment, 305
 - IDE Eclipse, 235
 - SWT, 235
 - Java ME, 235
 - Java SE, 235
 - Swing, 235
 - kontrolka ListView, 343
 - brak danych, 347
 - nagłówki sekcji, 352
 - obsługa zmian orientacji, 360
 - pusta lista, 348
 - stronicowanie, 344
 - wyświetlanie listy wierszy, 344
 - zachowywanie pozycji, 356
 - lista rozwijana, 251
 - Adapter, 251
 - getSelectedItem(), 252
 - kontrolka Spinner, 251
 - toString(), 252
 - menu, 316
 - addSubMenu(), 319
 - definicja, 316
 - niestandardowe menu, 317
 - obsługa wyboru opcji, 317
 - onCreateOptionsMenu(), 316, 319
 - onOptionsItemSelected(), 317, 320
 - początkowe menu, 320
 - podmenu, 318, 321
 - tworzenie, 316
 - wyświetlanie, 316
 - niestandardowe okno dialogowe, 328
 - Dialog, 328
 - okno O programie, 333, 336
 - okno ProgressDialog, 330

- run(), 330
 - z zakładkami, 331
- niestandardowy pasek tytułu, 275, 277
- obramowanie, 296
 - kolor, 297
 - kształt, 297
 - z zaokrąglonymi rogami, 298
- obrotowy mechanizm wybierania, 325
 - działanie, 325
 - kontrolki Android-Wheel, 325
- obsługa długiego kliknięcia, 253
 - setLongClickable(), 253
 - setOnLongClickListener(), 253
- obsługa zmian konfiguracji, 238
 - refreshUI(), 240
 - zachowywanie stanu aplikacji, 240
- otwieranie nowego ekranu, 283
 - kontrolka Button, 287
 - kontrolka TextView, 286
 - następne okno aplikacji, 290
 - pierwszy ekran, 290
 - warunki, 285
 - widżet aplikacji, 284
- pole wyboru, 247
- przechwytywanie klawiszy, 264
 - onKeyDown, 264
- przekształcanie pól tekstowych, 260
- przełączanie aktywności, 273
- przyciski
 - graficzne, 249
 - opcji, 247
 - tworzenie, 241
 - wysyłania, 262
- stosowanie kontrolerek, 246
 - RadioGroup, 247
 - Spinner, 246
 - ViewGroup, 247
- TableLayout, 63
- tworzenie przycisku, 241
 - onCreate(), 241
 - setOnClickListener(), 241
- widżet, 236
- widżet aplikacji, 236, 311
- wykrywanie gestów, 299
 - GestureDetector, 299
 - onTouchEvent, 299
- wyświetlanie pól tekstowych, 254
 - addTextChangedListener(), 256
 - afterTextChanged(), 256
 - beforeTextChanged(), 256
 - EditText, 254, 255
 - onCreate(), 256
 - onTextChanged(), 256
 - TextView, 254

- zakrywanie innych komponentów, 292
 - DrawerButton, 294
 - SlidingDrawer, 293, 295
- zaznaczanie grup, 265
 - kontrolka RatingBar, 265
 - onRatingChanged, 266
 - RatingBar, 265

I

- IDE Eclipse, 29, 235
 - SWT, 235
- Inkscape, 209, 228
 - Document Properties, 212
 - Export Bitmap, 211, 213, 229
 - SVG, 210
- instrukcje
 - adb logcat, 125
 - create project, 21
 - generowane elementy, 22
 - lista argumentów, 22
 - Intent.createChooser, 151
 - Intent.putExtra(), 151
- intencja
- intencja rozgłoszeniowa, 428
- intencje, 146
 - e-mail
 - z poziomu widoku, 147
 - z załącznikami, 150
 - Intent, 146
 - robienie zdjęć, 198
 - włączenie Bluetootha, 532
 - wybieranie numeru telefonu, 424
- internacjonalizacja, 571
 - Locale, 571
 - łańcuchy znaków, 572
 - wyszukiwanie, 575
 - tekst aplikacji, 572
 - wersje regionalne, 574

J

- Java, 19
 - biblioteki
 - RGraph, 224
 - definiowanie nowej klasy, 288
 - HTML5, 566
 - kod mostu, 567
 - interfejsy API, 20
 - kod natywny, 552
 - metody formatujące, 278
 - obsługa wyjątków, 76
 - Exception, 77
 - IOException, 77

- Java
 - obsługa wyjątków
 - kontrolowane, 76
 - niekontrolowane, 76
 - przekształcanie wyjątków, 77
 - RuntimeException, 77
 - Throwable, 77
 - VMError, 77
 - pakiet JDK, 29
 - parser ROME, 446
 - środowisko IDE Eclipse, 29
 - języki programowania, 549
 - C, 549
 - kod natywny, 551, 553
 - Clojure, 549
 - Erlang, 549
 - F#, 549
 - Groovy, 549
 - HTML5, 566
 - Scala, 549
 - Scheme, 549
 - skryptowe, 556
 - aplikacja SL4A, 556
 - BeanShell, 556
 - JavaScript, 557
 - JRuby, 557
 - Lua, 556
 - Perl, 556
 - Python, 556
 - Tcl, 557
 - JSON, 406
- ## K
- keytool, 584
 - klasy
 - AboutBox, 333, 334
 - ActivityInstrumentationTestCase2, 124
 - Adapter, 251, 350
 - AddressOverlay, 495
 - AlertDialog, 71, 321
 - android.Activity, 35
 - android.app.Activity, 64
 - android.app.Application, 79
 - android.media.Camera, 200
 - AndroidPlot, 207
 - ArrayAdapter, 350
 - AsyncTask, 145, 161, 164
 - BaseAdapter, 358
 - BookRank, 445
 - BugSenseHandler, 130
 - ChoiceFormat, 282
 - CountDownTimer, 55
 - Cube, 190
 - CustomDialog, 331
 - DateFormat, 97
 - DateUtils, 99
 - DbAdapter, 396
 - DemoCharts, 363
 - Dialog, 328
 - EditText, 254
 - Exception, 77
 - FaceDetectionView, 374
 - File, 384
 - metody zwracające informacje, 384
 - FileSaver, 82
 - Formatter, 99
 - FragmentTestActivity, 306
 - Geocoder, 479
 - GestureDetector, 299
 - Handler, 167
 - IncomingCallInterceptor, 419
 - Intent, 146
 - IOException, 77
 - ItemizedOverlay, 490
 - java.util.Date, 98
 - KeyListener, 99
 - ListActivity, 349
 - LoadingScreen, 292
 - Locale, 571
 - MapTest, 481
 - MapView, 481
 - MD5, 450
 - MediaRecorder, 371
 - MetarItem, 498
 - MyContentProvider, 177
 - MyLocationListener, 475
 - MyLocationOverlay, 486
 - MyOverlay, 488
 - OutgoingCallInterceptor, 421
 - PackageInfo, 333
 - ProgressDialog, 164
 - RatingBar, 265
 - właściwości, 265
 - Runnable, 159
 - Runtime, 545
 - RuntimeException, 77
 - RuntimeLog, 131
 - kod, 132
 - ScaleBarOverlay, 513
 - SensorManager, 522
 - Service, 155
 - ServiceManager, 158
 - składowa, 243
 - SlidingDrawer, 292
 - SmsManager, 426

- SQLiteOpenHelper, 401
- StatFs, 390
- TelephonyManager, 430
- TestFragment, 307
- TextView, 254
- Throwable, 77
 - hierarchia, 77
- Thread, 159
- Throwable, 77
- TicTacToeActivity, 239
- TicTacToeGame, 240
- Time, 99
- TouchEvent, 194
- TrackService, 156
- URL, 442
- URLConnection, 442
- View, 64, 194
- WebSettings, 453
 - wewnętrzna, 242
 - anonimowa, 244
- komponenty obsługi, 145
- komunikacja, 145
 - dostawcy treści, 175, 178
 - pobieranie danych, 175
- e-mail, 147
 - Intent.createChooser, 151
 - z poziomu widoku, 147
 - z załącznikami, 150
- intencje, 146
 - e-mail, 147, 150
 - Intent, 146
- IPC, 179
 - zdalne usługi, 179
- komponenty obsługi, 145, 167
- komunikaty rozgłoszeniowe, 157
 - kod klasy odbiornika, 158
 - publikowanie rozgłaszanych zdarzeń, 157
 - rejestrowanie odbiornika, 157
 - ServiceManager, 158
 - tworzenie odbiornika, 157
- odbiorniki rozgłoszeniowe, 145
- operacje w tle, 160
 - AsyncTask, 161, 164
 - doInBackground(), 162
 - execute(), 163
 - onClickListener(), 163
 - onCreate(), 162
 - onCreateDialog(), 164
 - onItemClick(), 162
 - pobieranie receptur, 166
 - ProgressDialog, 164
 - setCancelable(), 164
- pobieranie danych, 152
 - finish(), 154
 - getIntent().getExtras().getString(), 151
 - setResult(), 154
 - z aktywności podrzędnej, 154
- podtrzymywanie działania usługi, 155
 - onBind(), 156
 - onCreate, 155
 - onStartCommand(), 156
 - Service, 155
 - TrackService, 156
- przesyłanie danych, 151
 - Intent.getExtras(), 153
 - Intent.putExtra(), 151
 - MySubActivity.finish(), 153
 - onActivityResult(), 153
 - z aktywności, 153
- używanie wątków, 159
 - AsyncTask, 161
 - Handler, 167
 - kolejka wątków, 167
 - onCreate(), 159, 162
 - onItemClick(), 162
 - przesyłanie komunikatów, 166
 - run(), 159
 - Runnable, 159
 - start(), 159
 - Thread, 159
- komunikat rozgłoszeniowy, 420
- komunikaty toast, 315, 336
- kontrolki, 59
 - Android-Wheel, 325
 - AutoCompleteTextView, 257
 - zapełnianie, 259
 - Button, 287
 - CheckBox, 246
 - EditText, 63, 64, 254, 255, 501
 - z hasłem, 261
 - GridView, 238
 - ImageSwitcher, 368
 - ListView, 293, 343, 388
 - brak danych, 347
 - DemoCharts, 363
 - dostosowanie zawartości, 357
 - getCount(), 347
 - getItem(), 347
 - getTag(), 345
 - getView(), 358
 - konfigurowanie, 345
 - ListActivity, 349
 - nagłówki sekcji, 352
 - notifyDataSetChanged(), 360
 - obsługa zmian orientacji, 360
 - onConfigure(), 363

kontrolki

- ListView
 - pusta lista, 348
 - setListAdapter(), 345
 - stronicowanie, 344
 - wyświetlanie listy wierszy, 344
 - z ikonami, 349
 - zachowywanie pozycji, 356
- RadioButton, 246
- RadioGroup, 247
- RatingBar, 265
- rozmieszczanie, 59
- ScrollWheel, 326
- Spinner, 246, 251
- TableRaw, 60
- TabView, 503
- TabWidget, 503
- TextView, 26, 64, 97, 238, 254, 286, 451
- Timepicker, 322
- ViewGroup, 247
- WebView, 452
 - modyfikacja wyglądu, 453
- WheelView, 325

L

- liczby, 277
 - formatowanie, 277
 - kody formatujące, 278
 - metody formatujące, 278
- LinkedList, 468
- Linuks, 550
 - polecenia, 550
- lokalizacja, 473, 571
 - geokodowanie, 479
 - Geocoder, 479
 - odwrotne, 480
 - określanie położenia użytkownika, 473
 - aktualizowanie lokalizacji, 474
 - fikcyjne współrzędne GPS, 477
 - informacje z GPS-a, 475
 - MyLocationListener, 475
 - onLocationChanged(), 475
 - pobieranie danych o lokalizacji, 474
 - podawanie fikcyjnej lokalizacji, 478
 - setMockLocation(), 477
 - tryb COARSE, 474
 - tryb FINE, 474

M

- mapy, 473
 - Google'a, 480
 - AddressOverlay, 495
 - aktualna lokalizacja, 486
 - createItem(), 491
 - disableMyLocation(), 486
 - draw(), 490
 - drawCircle(), 500
 - getCenter(), 493
 - invalidate(), 489
 - isRouteDisplayed(), 481
 - ItemizedOverlay, 490
 - ItemizedOverlayXXXdraw(), 497
 - kilka znaczników, 493
 - kliknięcie znacznika, 494
 - konfigurowanie urządzenia AVD, 480
 - kontrolka TabView, 503
 - lista kontrolna, 484
 - MapTest, 481
 - MapView, 481
 - MapView.onTouchEvent(), 506
 - MetarItem, 498
 - MetarItem::draw(), 498
 - MyLocationOverlay, 486
 - MyOverlay, 488
 - obsługa długich kliknięć, 505
 - onCreate(), 492
 - populate(), 492
 - rejestrwanie klucza, 483
 - size(), 491
 - TabSpec.setContent(), 503
 - tworzenie nowego projektu, 481
 - tworzenie warstwy, 495
 - wyszukiwanie lokalizacji, 501, 502
 - wyświetlanie ikony, 499
 - wyświetlanie warstwy, 495
 - zmiany trybu mapy, 496
 - znacznik lokalizacji, 487
 - znacznik zastępczy, 493
 - OpenStreetMap, 509
 - aktualizowanie lokalizacji, 516
 - obsługa dotknięć warstwy, 514
 - przesuwanie mapy, 518
 - ScaleBarOverlay, 513
 - skala, 513
 - tworzenie warstw, 511
 - zmiana lokalizacji, 519
- Monkey, 140
- MOTODEV Studio, 576

- multimedia, 367
 - konwersja mowy na tekst, 380
 - konwersja tekstu na mowę, 381
 - mechanizm wykrywania twarzy, 373
 - detectFaces(), 374
 - FaceDetectionView, 374
 - init(), 374
- odtworzenie filmów, 367
 - YouTube, 368
- odtworzenie plików muzycznych, 377
 - bez interakcji z użytkownikiem, 379
 - MediaControler, 377
 - MediaPlayer, 377
 - onCreate(), 377
 - onTouchEvent(), 378
 - porządkowanie zasobów odtwarzacza, 378
- rejestracja filmów, 371
 - initRecorder(), 372
 - MediaRecorder, 371
 - onCreate(), 371
 - surfaceCreated(), 373

N

- nagłówki sekcji, 352
- narzędzia
 - ADB, 37
 - AndEngine, 458
 - android, 21
 - Android Localizer, 575
 - Google Analytics, 90
 - keytool, 584
 - Monkey, 140
 - MOTODEV Studio, 576
 - ProGuard, 592
 - StrictMode, 139
- NinePatch, 221

O

- obsługa połączeń telefonicznych, 417
 - pobieranie danych, 430
 - określanie stanu telefonu, 431
 - TelephonyManager, 430
 - przechwytywanie połączeń przychodzących, 418, 420
 - IncomingCallInterceptor, 419
 - kod manifestu aplikacji, 419
 - kod przechwytyjący, 418
 - odbiornik rozgłoszeniowy, 418
 - przechwytywanie połączeń wychodzących, 421
 - abortBroadcast(), 423
 - getResultData(), 424

- klasa przechwytyjąca, 421
- OutgoingCallInterceptor, 421
- przechwycone połączenie, 423
 - setResultData(), 421, 423
- wiadomości SMS, 425
 - kod do wysyłania, 426
 - odbieranie, 428
 - odebrana wiadomość, 426
 - onReceive(), 428
 - sendMultipartTextMessage(), 426
 - sendTextMessage(), 426
 - SmsManager, 426
 - wysyłanie, 425, 429
 - wybieranie numeru, 424
- odbiorniki rozgłoszeniowe, 83, 145, 418
- odzyskiwanie, 345
- OpenGL ES, 185, 187, 455
 - grafika trójwymiarowa, 188
 - buffer.position(0), 191
 - D-pad, 192
 - obracanie sześcianu, 192
 - onDrawFrame, 190
 - onSurfaceChanged, 190
 - requestFocus(), 194
 - setFocusableInTouchMode(true), 194
 - wyświetlanie sześcianu, 190
- OpenMoko, 521
- OpenStreetMap, 473, 509
- optymalizowanie kodu, 593
- OSM, *Patrz* OpenStreetMap
- OTF, 186

P

- Paint.NET, 218
 - Canvas Dialog, 218
 - Colors, 218
- pakiet ADK, 21
 - android, 21
 - create project, 21
- pakiet JDK, 29
- pakiet SDK, 29
 - aktualizacja, 46
 - możliwe błędy, 49
 - instalacja, 31
 - przykładowe programy, 44
 - SDK Manager, 32, 47
 - okno komunikatów, 48
 - wtyczka ADT, 29
- parser danych, 446
- parser ROME, 446
- PNG, 209
- pojemność karty SD, 390

- połączenia powłoki, 545
- powiadomienia, 338
- ProGuard, 592
- protokół HTTP, 441
- protokół SPP, 531
- przycisk graficzny, 249

R

- RESTful, 441
- rozgłaszanie uporządkowane, 420

S

- sieci społecznościowe, 467
 - Facebook, 467
 - LinkedInem, 468
 - protokół HTTP, 467
 - obsługa kliknięcia, 468
 - pobieranie logo, 468
 - tworzenie przycisków graficznych, 468
 - Twitter, 467, 470
 - getTwitterTimeline(), 470
 - wczytywanie chronologicznej listy tweetów, 470
- sieci, 441
 - 3G, 441
 - EDGE, 441
 - GPRS, 441
 - protokół HTTP, 441
 - protokół HTTPS, 441
 - protokół SPP, 531
 - RESTful, 441, 442
 - społecznościowe, 467
 - Facebook, 467
 - LinkedInem, 468
 - protokół HTTP, 467
 - Twitter, 467, 470
 - standard HDP, 531
 - XML/SOAP, 441
- singleton, 79, 240
- standard HDP, 531
- standardowe rozgłaszanie, 420
- sterowanie, 539
 - kopiowanie tekstu, 543
 - getText(), 543
 - setText(), 543
 - menedżer aktywności, 547
 - połączenia powłoki, 545
 - exec(), 545
 - process.waitFor(), 545
 - Runtime, 545
 - uruchamianie, 546

- połączenie z siecią, 539
 - informacje o sieci, 540
- tryb dzwonka, 541
 - ustawianie, 541, 542
- ustawienia projektu, 540
- włączanie wibracji, 544
- wyświetlanie powiadomień, 544
 - ledARGB(), 544
 - notify(), 544

- stoper z odliczaniem wstecznym, 55
 - CountDownTimer, 55
 - onFinish(), 55
 - onTick(), 55
- StrictMode, 139
- stronicowanie, 344
- SVG, 210

Ś

- środowisko Eclipse, 24
 - parametry nowego projektu, 25
 - projekt testowy, 124
 - TextView, 26
 - tworzenie nowego projektu, 45
 - tworzenie projektu powiązanego, 39
- środowisko IDE Eclipse, 29

T

- testy, 111
 - awarie, 125
 - adb logcat, 125
 - findViewById(), 127
 - NPE, 126
 - BugSense, 129
 - BugSenseHandler, 130
 - setContentView(), 130
 - chmura, 121
 - cykl życia, 134
 - onDestroy(), 138
 - onPause(), 138
 - onRestart(), 138
 - onStop(), 138
 - rejestrowanie zdarzeń, 135
 - scenariusze, 138
 - uruchamianie aktywności, 135, 136
 - wstrzymywanie aktywności, 135, 137
 - debugowanie kodu, 128
 - dane wyjściowe, 128
 - Log.d(), 128
 - lokalny dziennik czasu wykonania
 - RuntimeLog, 131
 - lokalny dziennik czasu wykonania, 131

- Monkey, 140
 - stosowanie, 141
- projekt testowy, 122
 - ActivityInstrumentationTestCase2, 124
 - konfiguracja, 122
 - tworzenie, 122
 - w środowisku Eclipse, 124
 - w środowisku IntelliJ IDEA, 123
- sterowanie programowaniem, 111
- StrictMode, 139
- urządzenia AVD, 113
 - definiowanie, 115
 - konfigurowanie, 114
 - tworzenie nowego urządzenia, 115
 - uruchamianie nowego urządzenia, 117
 - właściwości urządzeń, 118
- Tipster, 57
 - AlertDialog, 71
 - android.app.Activity, 64
 - calculate(), 68
 - EditText, 63
 - findViewById(), 64
 - getCheckedRadioButtonId(), 71
 - OnCheckedChangeListener(), 66
 - OnKeyListener(), 67
 - requestFocus(), 65
 - reset(), 69
 - setEnabled(), 65
 - showErrorAlert(), 69, 71
 - TableLayout, 60,
 - TableRaw, 60
 - View, 64
- TTF, 186
- Twittera, 467
- tworzenie
 - bazy SQLite, 401
 - ekranów powitalnych, 84
 - ikony, 208, 215
 - kopii zapasowej, 102
 - menu, 316
 - nowego urządzenia AVD, 115
 - odbiornika rozgłoszeniowego, 157
 - projektu testowego, 122
 - przycisku, 241, 468
 - serwera, 536
 - warstw mapy, 495, 511
 - wykresów, 224

U

- urządzenia AVD, 23, 113
 - definiowanie, 115
 - konfigurowanie, 112, 114, 480

- opcje uruchomieniowe, 116, 119
- testy, 113
- tworzenie nowego urządzenia, 115
- uruchamianie nowego urządzenia, 117
- właściwości urządzeń, 118
- urządzenia GPS, 473

W

- wiadomości SMS, 426
 - odbieranie, 428
 - get(), 428
 - intencja rozgłoszeniowa, 428
 - onReceive(), 428
 - SmsManager, 426
 - wysyłanie, 425, 429
 - kod, 426
 - okno Emulator Control, 430
 - sendMultipartTextMessage(), 426
 - sendTextMessage(), 426
- widżet, 236
- widżet aplikacji, 236, 284, 311
- wtyczka ADT, 24, 29
 - instalacja, 33
- wrażenia regularne, 444
- wyświetlanie
 - czasu i daty, 97
 - danych, 207
 - ikony, 499
 - listy wierszy, 344
 - menu, 316
 - powiadomień, 338, 544
 - pól tekstowych, 254
 - reklam, 588
 - stron internetowych, 452
 - warstwy, 495
 - zawartości katalogu, 387

X

- XML/SOAP, 441

Y

- YouTube, 367

Z

- zaciemnianie kodu, 593

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

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/testsellery>

Zamów informacje o nowościach:

📍 <http://helion.pl/nowości>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 03

e-mail: helion@helion.pl

<http://helion.pl>



ISBN 978-83-246-6269-2



Cena 89,00 zł