



Wydanie IV

Carmen Delessio, Lauren Darcey, Shane Conder

Android Studio w 24 godziny

Wygodne programowanie dla platformy Android

SAMS

Helion 

Tytuł oryginału: Sams Teach Yourself Android™ Application Development in 24 Hours, Fourth Edition

Tłumaczenie: Grzegorz Kowalczyk
Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

ISBN: 978-83-283-2150-2

Authorized translation from the English language edition: ANDROID APPLICATION DEVELOPMENT IN 24 HOURS, SAMS TEACH YOURSELF, ISBN 0672337398; by Carmen Delessio; and Lauren Darcey; and Shane Conder; published by Pearson Education, Inc, publishing as SAMS Publishing.
Copyright © 2016 by Carmen Delessio, Lauren Darcey, and Shane Conder.

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 Pearson Education Inc.

Polish language edition published by HELION S.A. Copyright © 2016.

Several images in this book use scenes from the online movie Big BuckBunny to illustrate the use of online video and using a VideoView control. This movie and related material is distributed under a Creative Commons license. For more information on the movie, go to <http://www.bigbuckbunny.org/>.

Blender Foundation | <http://www.blender.org>

Copyright © 2008, Blender Foundation / <http://www.bigbuckbunny.org>

Some images in this book are reproduced or are modifications based on work created and shared by Google and used according to terms described in the Creative Commons 3.0 Attribution License.

The following are registered trademarks of Google:

Android, Google Play, Android TV, Android Wear, Google, and the Google logo are registered trademarks of Google Inc., and are used here with permission.

Flickr and Flickr API are registered trademarks of Yahoo!.

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock Images LLC.

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.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą 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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem: <ftp://ftp.helion.pl/przyklady/as24w4.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/as24w4>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

O autorach	15
Wstęp	17
Co nowego w tym wydaniu	17
Dla kogo przeznaczona jest ta książka	18
Jak ta książka jest zorganizowana	18
Kody źródłowe przykładowych aplikacji	19
Część I Podstawowe zagadnienia związane z systemem Android	21
Godzina 1. Wprowadzenie do systemu Android	23
Podstawowe pojęcia związane z systemem Android	23
Podstawowe pojęcia związane z tworzeniem aplikacji	26
Rozpoczynamy pracę z pakietem Android Studio	32
Podsumowanie	36
Pytania i odpowiedzi	36
Warsztaty	37
Ćwiczenia	37
Godzina 2. Intencje	39
Zastosowanie intencji do uruchamiania aktywności	39
Zastosowanie intencji niejawnych	51
Obsługa intencji niejawnych	57
Podsumowanie	59
Pytania i odpowiedzi	59
Warsztaty	60
Ćwiczenia	60

Godzina 3.	Zasoby aplikacji	61
	Zasoby w Twoim projekcie	61
	Zastosowanie wspólnych zasobów aplikacji	65
	Korzystanie z alternatywnych zasobów aplikacji	74
	Internacjonalizacja aplikacji — korzystanie z zasobów dla innych wersji językowych	76
	Podsumowanie	78
	Pytania i odpowiedzi	79
	Warsztaty	79
	Ćwiczenia	79
Godzina 4.	Aktywności i fragmenty	81
	Praca z aktywnościami	81
	Cykl życia aktywności	93
	Wprowadzenie do fragmentów	97
	Podsumowanie	102
	Pytania i odpowiedzi	102
	Warsztaty	103
	Ćwiczenia	103
Godzina 5.	Responsywność aplikacji — działanie w tle	105
	Praca w tle	105
	Zastosowanie zadań asynchronicznych AsyncTask	109
	Usługi Service oraz IntentService	112
	Podsumowanie	121
	Pytania i odpowiedzi	121
	Warsztaty	122
	Ćwiczenia	122
Część II	Tworzenie interfejsów użytkownika	123
Godzina 6.	Korzystanie z podstawowych formantów interfejsu użytkownika	125
	Korzystanie z palety formantów Android Studio	125
	Obsługiwanie danych wprowadzanych przez użytkownika	127
	Zastosowanie przycisków do uruchamiania akcji	132
	Podsumowanie	138
	Pytania i odpowiedzi	138
	Warsztaty	138
	Ćwiczenia	139

Godzina 7.	Tworzenie układów interfejsów użytkownika	141
	Rozpoczynamy pracę z układami interfejsów użytkownika	141
	RelativeLayout dla zaawansowanych	145
	Wspólne atrybuty	149
	Inne rodzaje układów	150
	Podsumowanie	153
	Pytania i odpowiedzi	153
	Warsztaty	153
	Ćwiczenia	154
Godzina 8.	Kontenery ListView i adaptery	155
	Rozpoczynamy pracę z kontenerami ListView	155
	Rozszerzanie klasy BaseAdapter	160
	Zastosowanie wzorca ViewHolder	167
	Podsumowanie	172
	Pytania i odpowiedzi	172
	Warsztaty	172
	Ćwiczenia	173
Godzina 9.	Interfejs Material Design	175
	Ewolucja w projektowaniu aplikacji	175
	Wprowadzenie do interfejsu Material Design	177
	Implementacja interfejsów Material Design	182
	Podsumowanie	197
	Pytania i odpowiedzi	197
	Warsztaty	197
	Ćwiczenia	198
Godzina 10.	Inne widoki i formanty	199
	Formanty przeznaczone do pobierania informacji	199
	Wyświetlanie postępu realizacji zadania	204
	Wyświetlanie danych	208
	Inne widoki	212
	Podsumowanie	215
	Pytania i odpowiedzi	215
	Warsztaty	216
	Ćwiczenia	216

Godzina 11.	Widoki ImageView i bitmapy	217
	Praca z widokami ImageView	217
	Praca z obiektami klasy Bitmap i klasy Canvas	224
	Zastosowanie biblioteki Picasso	230
	Podsumowanie	232
	Pytania i odpowiedzi	232
	Warsztaty	232
	Ćwiczenia	233
Godzina 12.	Widoki VideoView i odtwarzanie mediów	235
	Odtwarzanie plików wideo	235
	Obsługa zdarzeń widoku VideoView	239
	Odtwarzanie plików audio za pomocą obiektów MediaPlayer	243
	Inne klasy związane z obsługą multimediiów	245
	Podsumowanie	245
	Pytania i odpowiedzi	246
	Warsztaty	246
	Ćwiczenia	247
Godzina 13.	Nawigowanie w aplikacji	249
	Zastosowanie paska ActionBar	249
	Zastosowanie paska Toolbar	256
	Zastosowanie wysuwanych paneli menu	261
	Podsumowanie	271
	Pytania i odpowiedzi	271
	Warsztaty	272
	Ćwiczenia	272
Część III	Praca z danymi	273
Godzina 14.	Korzystanie z systemu plików	275
	Omówienie systemu plików	275
	Zapisywanie prywatnych danych aplikacji	279
	Zapisywanie danych w katalogach publicznych	283
	Podsumowanie	285
	Pytania i odpowiedzi	286
	Warsztaty	286
	Ćwiczenia	286

Godzina 15.	Zastosowanie klasy SharedPreferences	287
	Zastosowanie klasy SharedPreferences do przechowywania danych	287
	Zapisywanie preferencji użytkownika	291
	Podsumowanie	301
	Pytania i odpowiedzi	301
	Warsztaty	301
	Ćwiczenia	302
Godzina 16.	Zapisywanie danych w bazie SQLite	303
	Tworzenie baz danych z tabelami	303
	Zarządzanie danymi przy użyciu klasy SQLiteOpenHelper	305
	Dodawanie, usuwanie i aktualizacja danych	308
	Zapytania danych i zastosowanie kursorów	310
	Korzystanie z baz danych w aplikacjach	312
	Podsumowanie	316
	Pytania i odpowiedzi	317
	Warsztaty	317
	Ćwiczenia	317
Godzina 17.	Dostęp do chmury: praca ze zdalnym interfejsem API	319
	Pobieranie zdalnych danych	320
	Pobieranie i parsowanie danych zapisanych w formacie JSON	325
	Wszystko razem, czyli tworzymy prostą aplikację	330
	Sprawdzanie połączenia sieciowego	334
	Podsumowanie	338
	Pytania i odpowiedzi	338
	Warsztaty	338
	Ćwiczenia	339
Godzina 18.	Wprowadzenie do pracy z dostawcami treści	341
	Wprowadzenie do pracy z dostawcami treści	341
	Wszystko o kalendarzu	342
	Pobieranie danych z kalendarza za pośrednictwem dostawcy treści	344
	Podsumowanie	352
	Pytania i odpowiedzi	352
	Warsztaty	352
	Ćwiczenia	352

Godzina 19.	Tworzenie dostawców treści	353
	Tworzenie URI dla pobierania danych	353
	Zastosowanie adaptera PieDbAdapter	354
	Tworzenie dostawcy treści	354
	Zastosowanie dostawcy treści MyContentProvider w aplikacji	362
	Podsumowanie	364
	Pytania i odpowiedzi	364
	Warsztaty	365
	Ćwiczenia	365
Godzina 20.	Loadery i adaptory CursorAdapter	367
	Jak działają loadery?	367
	Klasy loaderów	368
	Stany loadera	369
	Tworzenie adapterów klasy CursorAdapter	374
	Pozostała część aplikacji	377
	Podsumowanie	378
	Pytania i odpowiedzi	378
	Warsztaty	378
	Ćwiczenia	379
Część IV	Idziemy dalej	381
Godzina 21.	Używanie powiadomień	383
	Wprowadzenie do powiadomień	383
	Tworzenie powiadomień i zarządzanie nimi	385
	Dostosowywanie powiadomień	390
	Podsumowanie	392
	Pytania i odpowiedzi	392
	Warsztaty	392
	Ćwiczenia	393
Godzina 22.	Aplikacje dla Android TV i urządzeń typu Android Wear	395
	Android jako platforma	395
	Projektowanie aplikacji dla urządzeń Android Wear	396
	Tworzenie aplikacji dla Android TV	403
	Podsumowanie	407

Pytania i odpowiedzi	407
Warsztaty	408
Ćwiczenia	408
Godzina 23. Inne komponenty platformy Android	409
Używanie usług Google Play	409
Używanie usług Google Play Services do lokalizacji	412
Używanie bibliotek open source i zewnętrznych pakietów SDK	417
Zaglądamy głębiej do systemu Android	418
Podsumowanie	423
Pytania i odpowiedzi	423
Warsztaty	423
Ćwiczenia	424
Godzina 24. Publikowanie aplikacji	425
Przygotowanie aplikacji do opublikowania	425
Udostępnianie aplikacji światu	433
Zarabianie na aplikacjach	435
Podsumowanie	436
Pytania i odpowiedzi	436
Warsztaty	436
Ćwiczenia	437
 Skorowidz	 439

Godzina 7

Tworzenie układów interfejsów użytkownika

W tej godzinie:

- ▶ Rozpoczynamy pracę z układami interfejsów użytkownika
- ▶ Zastosowanie układu `RelativeLayout`
- ▶ Zastosowanie różnych rodzajów układów interfejsu
- ▶ Tworzenie efektywnych układów interfejsu użytkownika

W systemie Android układy (ang. *layouts*) są wykorzystywane do prawidłowego wyświetlania interfejsu użytkownika na różnych urządzeniach. Każda aktywność, którą utworzyłeś w aplikacji, posiada powiązany z nią plik XML układu interfejsu, wykorzystujący komponent `RelativeLayout`. Pliki układu to szablony opisujące wygląd ekranu aplikacji. W tej godzinie poznasz różne rodzaje układów interfejsów użytkownika i dowiesz się, jak ich używać.

Rozpoczynamy pracę z układami interfejsów użytkownika

Pliki układu są wykorzystywane do opisywania tego, jak będzie wyglądał ekran aplikacji (lub jego fragment), lub inaczej mówiąc, jak będzie wyglądał interfejs użytkownika aplikacji. W plikach układu przechowywane są definicje formantów interfejsu użytkownika, takich jak na przykład `EditText` czy `Button`. Poszczególne formanty interfejsu użytkownika mogą odwoływać się do innych zasobów, takich jak ciągi znaków, kolory, wymiary czy komponenty typu *drawable*.

W godzinie 6. „Korzystanie z podstawowych formantów interfejsu użytkownika” dowiedziałeś się, że widoki to najbardziej podstawowe elementy składowe interfejsów użytkownika w systemie Android. Każdy formant, taki jak `Button`, `TextView` czy `Layout`, to rodzaj widoku. Obiekty `ViewGroup` to kontenery dla innych widoków. Kiedy przy użyciu Android Studio umieszczasz w interfejsie użytkownika przycisk `Button`, to tym samym dodajesz przycisk `Button` do układu `RelativeLayout`. W takim przypadku przycisk `Button` jest traktowany jako widok układu `RelativeLayout`.

Oprócz układu `RelativeLayout` istnieje również kilka innych rodzajów układów, takich jak `LinearLayout`, `FrameLayout`, `TableLayout` czy `GridLayout`.

Każda klasa układu jest kontenerem dla innych widoków. W poszczególnych klasach układów obowiązują różne reguły dodawania widoków podrzędnych (formantów). Atrybuty widoków podrzędnych często zależą od tego, w jakim układzie został umieszczony dany widok. Jeśli na przykład korzystasz z układu `RelativeLayout`, to możesz umieszczać przyciski `Button` jeden pod drugim, używając atrybutu `android:layout_below`. Ten atrybut nie jest używany w układach `LinearLayout` czy `FrameLayout`.

Używanie plików XML układów do definiowania widoków dla interfejsu użytkownika jest powszechnie stosowaną praktyką. Widoki możesz również tworzyć w sposób programowy. Układ zawsze spełnia rolę kontenera dla innych widoków, niezależnie od tego, czy został zdefiniowany w pliku XML, czy też został utworzony dynamicznie w czasie działania programu.

Jak się już zapewne zdążyłeś przekonać, pliki układów są przechowywane w odpowiedniej strukturze katalogu `/res/layout`.

Uwaga Uwaga

Zastosowanie znacznika `include` w pliku układu

W plikach XML opisujących układ interfejsu użytkownika możemy używać znacznika `<include />`, pozwalającego na dołączanie innych układów do układu macierzystego. Aby skorzystać z takiego znacznika, powinieneś najpierw utworzyć plik układu pomocniczego, który osadzany jest w innych układach, a następnie w kodzie XML pliku możesz odwołać się do niego tak, jak się odwołujemy do innych zasobów. Na przykład: aby w danym pliku XML odwołać się do pliku układu o nazwie `basicHeader.xml`, powinieneś użyć znacznika `<include layout="@layout/basicHeader"/>`.

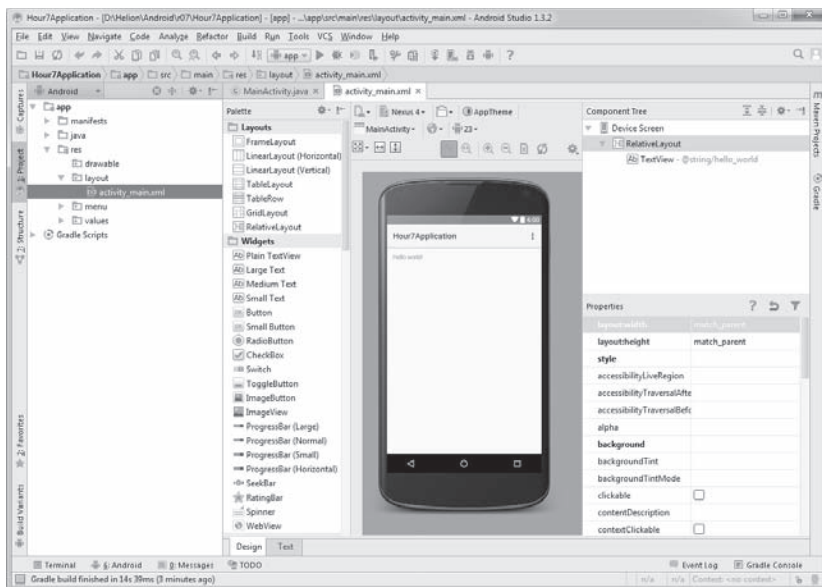
Projektowanie układów w Android Studio

Kiedy przy użyciu Android Studio tworzysz nową aktywność, automatycznie tworzony jest powiązany z nią plik układu wykorzystujący kontener `RelativeLayout`, w którym domyślnie tworzony jest pojedynczy formant `TextView` będący widokiem podrzędnym kontenera `RelativeLayout`.

Na listingu 7.1 przedstawiono początkową, domyślną zawartość pliku `activity_main.xml`. Zauważ, że nie ma w nim żadnej informacji wskazującej, gdzie formant `TextView` powinien zostać umieszczony. W układzie `RelativeLayout` domyślną pozycją widoku podrzędnego jest lewy górny róg obszaru układu. Na rysunku 7.1 przedstawiono formant `TextView` umieszczony w takiej domyślnej lokalizacji.

Listing 7.1. Plik XML z układem RelativeLayout (activity_main.xml)

```
1: <RelativeLayout xmlns:android="http://schemas.android.com/
   ↳apk/res/android"
2:     xmlns:tools="http://schemas.android.com/tools"
3:     android:layout_width="match_parent"
4:     android:layout_height="match_parent"
5:     android:paddingLeft="@dimen/activity_horizontal_margin"
6:     android:paddingRight="@dimen/activity_horizontal_margin"
7:     android:paddingTop="@dimen/activity_vertical_margin"
8:     android:paddingBottom="@dimen/activity_vertical_margin"
9:     tools:context=".MainActivity">
10:
11:     <TextView android:text="@string/hello_world"
12:         android:layout_width="wrap_content"
13:         android:layout_height="wrap_content" />
14:
15: </RelativeLayout>
```

**RYSUNEK 7.1.** Układ RelativeLayout w Android Studio

W Android Studio możesz edytować pliki w trybie wizualnym (karta *Design*) bądź w trybie tekstowym (karta *Text*). Widoczny na rysunku 7.1 edytor układu pracuje w trybie wizualnym (zwróć uwagę na karty *Design* i *Text* w dolnej części okna).

Edytowanie plików układu bezpośrednio w trybie tekstowym może być bardzo użyteczne. Używając układu `RelativeLayout`, rozmieszczasz widoki podrzędne względem innych widoków lub względem krawędzi kontenera nadrzędnego. Nowy widok może zostać umieszczony powyżej, poniżej, z lewej lub z prawej

strony innego widoku. Często okazuje się, że w takiej sytuacji łatwiej będzie rozmieszczać elementy interfejsu, edytując plik XML bezpośrednio w trybie tekstowym, niż korzystając z edytora wizualnego.

W praktyce zazwyczaj będziemy pracować w trybie mieszanym, przełączając w miarę potrzeb edytor układów Android Studio na przemian do trybu tekstowego i do trybu wizualnego.

Tworzenie układów bezpośrednio w języku XML

W razie potrzeby możesz bezpośrednio edytować pliki XML opisujące układ elementów interfejsu. W miarę jak będziesz nabierał doświadczenia, tworząc kolejne aplikacje, bezpośrednie edytowanie plików XML będzie coraz bardziej naturalnym procesem. Poprzez częste przełączanie Android Studio do widoku XML szybko zorientujesz się, jaki kod XML jest generowany poprzez poszczególne rodzaje formantów. Kiedy zmieniasz właściwości wybranego formantu z poziomu edytora wizualnego, odpowiednie zmiany są automatycznie wprowadzane w pliku XML. Po nabraniu doświadczenia i praktyki w tworzeniu interfejsów użytkownika wprowadzanie odpowiednich zmian w plikach XML opisujących układ stanie się po prostu szybsze i wygodniejsze.

Ponieważ pliki XML możesz edytować jak zwykłe pliki tekstowe, możesz korzystać z takich udogodnień jak wyszukiwanie określonych fraz i znaczników czy szybkie wprowadzanie zmian w odpowiednich miejscach.

Korzystanie z zasobów układu z poziomu kodu programu

W godzinie 6. budowaliśmy interfejs użytkownika składający się z formantów `TextView`, `Button` i `EditText`. Poszczególne widoki były zdefiniowane w pliku `activity_main.xml`, zlokalizowanym w katalogu `/res/layout`. W kodzie XML definicje formantów zostały umieszczone wewnątrz elementu `RelativeLayout`.

Aby użyć takiego pliku układu w danej aktywności, musisz wywołać metodę `setContentView()`. Wiersz kodu przedstawiony poniżej jest automatycznie dołączany do kodu aktywności podczas generowania projektu.

```
setContentView(R.layout.activity_main);
```

Aktywność wykorzystuje metodę `setContentView()` do powiązania aktywności z układem. Kiedy chcesz uzyskać dostęp do formantów zdefiniowanych w pliku układu, musisz użyć metody `findViewById()`. Aby utworzyć nowy obiekt klasy `Button` reprezentujący przycisk o identyfikatorze `id`, zdefiniowany w pliku układu, a następnie przypisać go do zmiennej `myButton`, powinieneś użyć następującego wiersza kodu:

```
Button myButton = (Button) findViewById(R.id.button1);
```

Metoda `findViewById()` jest dostępna zarówno z poziomu aktywności, jak i z poziomu dowolnego widoku. Widoki możesz tworzyć dynamicznie z układów XML. Plik układu jest rozwijany do obiektu `View` za pomocą klasy `LayoutInflater`.

Po rozwinięciu układu możesz korzystać z jego formantów. Na listingu zamieszczonym poniżej przedstawiono przykład zastosowania klasy `LayoutInflater` i tworzenia widoku:

```
LayoutInflater inflater = LayoutInflater.from(context);
View exampleView = inflater.inflate(R.layout.example, container, false);
Button myExampleButton = (Button)exampleView.findViewById(R.id.button1);
```

RelativeLayout dla zaawansowanych

Układ `RelativeLayout` zapewnia dużą kontrolę nad rozmieszczaniem widoków podrzędnych oraz elastyczność pozwalające na projektowanie interfejsów użytkownika efektywnie działających na urządzeniach o różnych wielkościach ekranów.

W układzie `RelativeLayout` poszczególne widoki podrzędne są rozmieszczane względem innych widoków lub względem krawędzi kontenera nadrzędnego.

Wyrównywanie do krawędzi kontenera nadrzędnego

Kiedy widok podrzędny jest rozmieszczany w odniesieniu do kontenera nadrzędnego, może być wyśrodkowany, położony względem krawędzi kontenera bądź położony względem początku lub końca kontenera.

Jeżeli widok jest położony względem krawędzi kontenera, możesz używać takich atrybutów jak `layout_alignParentLeft` i `layoutAlignParentStart`. Różnica pomiędzy tymi atrybutami polega na tym, że ten drugi atrybut jest przeznaczony do obsługi języków, w których tekst jest zapisywany od prawej do lewej (ang. *right-to-left languages*; *RTL languages*). W językach RTL pisanie tekstu rozpoczyna się od prawego marginesu. Jeżeli użyjesz atrybutu `layoutAlignParentStart`, widok zostanie umieszczony przy prawej krawędzi ekranu.

Poniżej przedstawiamy zestawienie atrybutów, które są używane do pozycjonowania widoków podrzędnych względem krawędzi kontenera nadrzędnego:

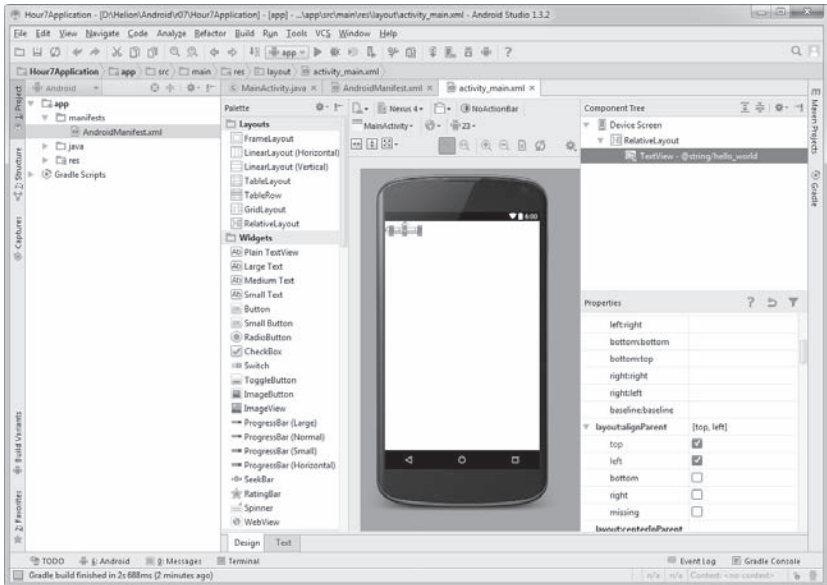
- ▶ `android:layout_alignParentStart`
- ▶ `android:layout_alignParentEnd`
- ▶ `android:layout_alignParentBottom`
- ▶ `android:layout_alignParentRight`
- ▶ `android:layout_alignParentTop`
- ▶ `android:layout_alignParentLeft`

W kolejnym zestawieniu przedstawiamy atrybuty, które są używane do pozycjonowania widoków podrzędnych względem środka kontenera nadrzędnego:

- ▶ `android:layout_centerInParent`
- ▶ `android:layout_centerVertical`
- ▶ `android:layout_centerHorizontal`

Na rysunku 7.2 przedstawiono formant `TextView` wyrównany do lewej, górnej krawędzi kontenera. W pliku układu nasz formant został opisany w następujący sposób:

```
<TextView android:text="@string/hello_world"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
android:layout_alignParentLeft="true" />
```



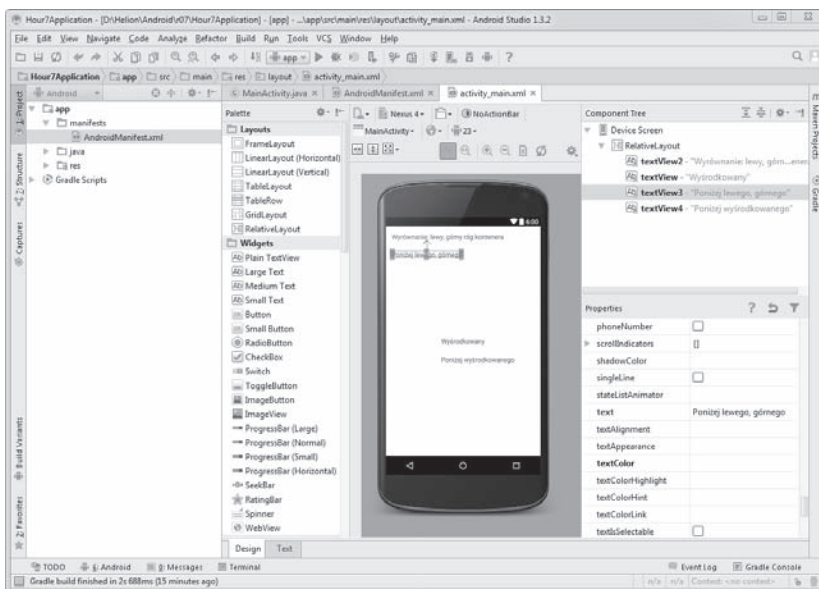
RYSUNEK 7.2. Wyrównywanie formantu `TextView` do kontenera nadrzędnego

Wyrównywanie widoków względem siebie

Widoki podrzędne mogą być umieszczane powyżej, poniżej, po prawej lub po lewej stronie innych widoków. Na ekranie musi się znajdować co najmniej jeden widok, który będzie spełniał rolę punktu odniesienia. Bardzo często zdarza się, że najpierw jeden bądź więcej widoków zostaje wyrównane względem kontenera, a następnie pozostałe widoki są rozmieszczane względem tych widoków bazowych.

Podobnie jak dzieje się w przypadku wyrównywania względem kontenera nadrzędnego, klasa `RelativeLayout` również posiada atrybuty pozwalające na odpowiednie pozycjonowanie dla języków RTL. Na przykład zamiast atrybutu `android:layout_toLeftOf` możesz w takiej sytuacji używać atrybutu `android:layout_toStartOf`.

Na rysunku 7.3 dwa formanty `TextView` — pierwszy znajdujący się w lewym górnym rogu ekranu oraz drugi zlokalizowany centralnie na środku ekranu — wyrównane są do kontenera nadrzędnego. Pozostałe dwa formanty są rozmieszczone względem tych dwóch formantów bazowych.



RYSUNEK 7.3. Wyrównywanie formantów TextView względem siebie

Zawartość pliku XML odpowiadająca układowi przedstawionemu na rysunku 7.3 wygląda następująco:

```
<RelativeLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView android:text="@string/center" android:
    layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:layout_centerInParent="true"
        android:id="@+id/textViewCenter" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="@string/top_left"
        android:id="@+id/textViewTopLeft"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true" />
```

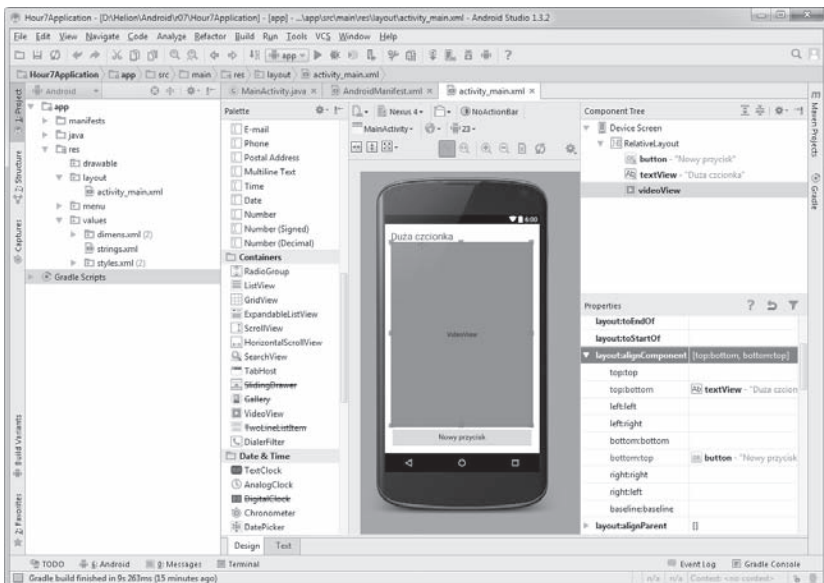
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/below_top_left"
    android:id="@+id/textViewBelowTopLeft"
    android:layout_below="@+id/textViewTopLeft" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/below_center"
    android:id="@+id/textViewBelowCenter"
    android:layout_below="@+id/textViewCenter"
    android:layout_alignStart="@+id/textViewCenter" />
```

```
</RelativeLayout>
```

Jednym ze sposobów wykorzystania układu RelativeLayout jest umieszczenie w środku interfejsu widoku, który odwołuje się do czegoś znajdującego się nad nim i pod nim. Na rysunku 7.4 przedstawiono układ interfejsu w Android Studio, w którym znajdują się następujące elementy:

- ▶ Przycisk Button wyrównany do dolnej krawędzi kontenera:
android:layout_alignParentBottom="true"
- ▶ Pole TextView umieszczone w lewym górnym rogu.
- ▶ Formant VideoView, który znajduje się poniżej pola TextView i powyżej przycisku Button.



RYSUNEK 7.4. Projekt z wykorzystaniem układu RelativeLayout

Aby ustawić pozycję formantu `VideoView`, powinieneś użyć następujących właściwości:

```
android:layout_below="@+id/textView1"  
android:layout_above="@+id/loadPhotosButton"
```

Wspólne atrybuty

Niektóre atrybuty są wspólne dla układów i umieszczonych w nich widoków podrzędnych. Na przykład wszystkie układy posiadają właściwości `android:layout_width` i `android:layout_height`, wykorzystywane do kontrolowania odpowiednio szerokości i wysokości elementów. Wartości takich atrybutów mogą być wyrażane zarówno w pikselach logicznych niezależnych od gęstości ekranu (na przykład `20dp`), jak i w postaci predefiniowanych stałych, posiadających określone znaczenie (na przykład `match_parent` czy `wrap_content`).

Stała `match_parent` powoduje, że dany widok zostanie przeskalowany do rozmiarów układu nadrzędnego, a stała `wrap_content` powoduje, że rozmiary widoku zostają dostosowane do zawartości elementu.

Marginesy układów

Marginesy układów określają odległość pomiędzy widokiem podrzędnym a krawędzią kontenera nadrzędnego. Aby przesunąć przycisk w dół i w prawo, musisz dla niego określić margines od górnej i od lewej krawędzi układu. Na przykład dodając do definicji przycisku w pliku układu dwa wiersze zamieszczone poniżej, przesuń go o 40 pikseli w dół i 120 pikseli w prawo.

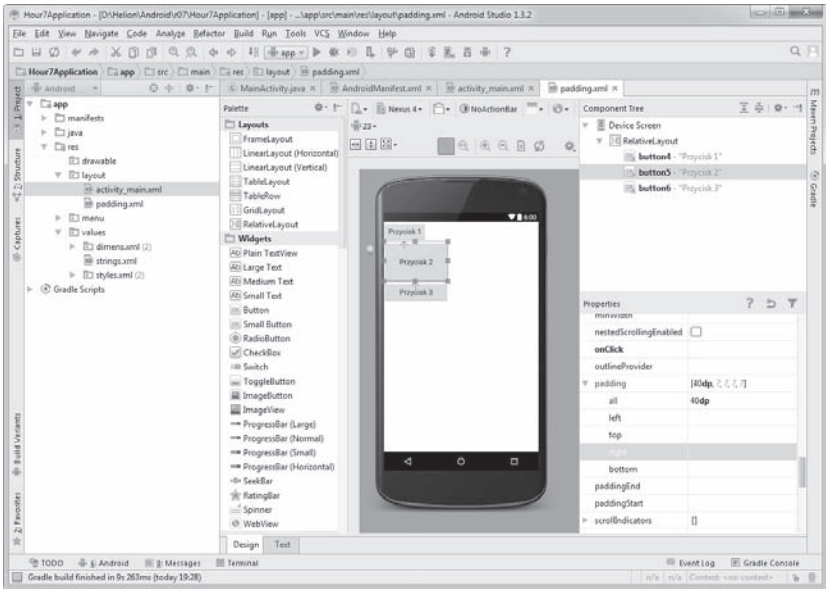
```
android:layout_marginTop="40dp"  
android:layout_marginLeft="120dp"
```

Właściwość padding

`Padding` to ilość miejsca dodawanego z wybranej strony formantu po to, aby przesunąć jego zawartość. Właściwość `padding` odnosi się do wewnętrznych marginesów formantu. W przypadku przycisków `Button` będzie to dodatkowa przestrzeń dodawana pomiędzy tekstem etykiety a krawędziami przycisku. `Padding` możesz ustawić dla całego formantu lub dla każdej krawędzi osobno.

Na rysunku 7.5 przedstawiono nowy układ XML z trzema przyciskami, ilustrujący zastosowanie `padding`.

Dla pierwszego przycisku właściwość `padding` nie została ustawiona. Drugi przycisk ma właściwość `padding` ustawioną na wartość `40dp` ze wszystkich stron. Możesz to zrobić za pomocą polecenia `android:padding="40dp"`. Dzięki takiemu poleceniu z każdej strony przycisku zostanie dodane 40 pikseli `dp`. Trzeci przycisk ma właściwość `padding` ustawioną na `40dp` z lewej i z prawej strony. Prawy `padding` został ustawiony za pomocą polecenia `android:padding` ↪ `Right="40dp"`, a lewy `padding` za pomocą polecenia `android:padding` ↪ `Left="40dp"`.



RYSUNEK 7.5. Zastosowanie paddingu

Inne rodzaje układów

Powszechnie stosowanym rozwiązaniem jest zawieranie całego ekranu aplikacji w jednym, dużym układzie nadrzędnym. Najczęściej wykorzystywane są układy `RelativeLayout` i `LinearLayout`. W tabeli 7.1 przedstawiono zestawienie najczęściej wykorzystywanych układów.

TABELA 7.1. Najpopularniejsze formanty układów

Nazwa formantu układu	Opis	Najważniejsze atrybuty
<code>LinearLayout</code>	Układ liniowy, w którym każdy kolejny widok podrzędny jest umieszczany za poprzednim, w jednym wierszu bądź w jednej kolumnie.	Orientacja (pionowa lub pozioma).
<code>RelativeLayout</code>	Układ względny, w którym każdy kolejny widok podrzędny jest pozycjonowany względem innych widoków lub względem krawędzi układu nadrzędnego.	Wiele atrybutów wyrównania odpowiadających za położenie widoku podrzędnego względem innych widoków.

TABELA 7.1. Najpopularniejsze formanty układów — ciąg dalszy

Nazwa formantu układu	Opis	Najważniejsze atrybuty
FrameLayout	Układ ramowy, w którym kolejne widoki podrzędne są ustawiane jeden na drugim i tworzą stos.	Kolejność umieszczania widoków podrzędnych jest bardzo istotna. Poszczególne widoki mogą się na siebie nakładać.
TableLayout	Układ tabeli, w którym poszczególne widoki podrzędne są rozmieszczane w wierszach i kolumnach (tak jak w tabeli).	Każdy wiersz wymaga elementu TableRow.
GridLayout	Układ rastrowy, w którym poszczególne widoki podrzędne są umieszczane w prostokątnych obszarach tworzących raster.	Wiersze i kolumny z atrybutami RowSpec i ColumnSpec.

Zastosowanie układu LinearLayout

W układach LinearLayout widoki rozmieszczane są w wierszach lub kolumnach. Kierunek rozmieszczania widoków jest określany za pomocą atrybutu `orientation`. Jeżeli korzystamy z orientacji pionowej, widoki podrzędne są rozmieszczane w kolumnie jeden pod drugim. Jeżeli korzystamy z orientacji poziomej, widoki podrzędne rozmieszczane są w wierszu obok siebie. Choć układy posiadają bardzo wiele wspólnych atrybutów, najważniejsze różnice między nimi polegają na kolejności i regułach rozmieszczania formantów na ekranie. Atrybut `orientation` jest unikatowy dla układu LinearLayout:

```
android:orientation="vertical"
```

Właściwość gravity

Widoki podrzędne posiadają właściwość `layout_gravity`, która może przyjmować wartości takie jak `left`, `right`, `center` i tak dalej:

```
android:layout_gravity="right"
```

Właściwość `gravity` odnosi się do widoków podrzędnych, ale może być także używana dla całych układów. Na przykład poprzez ustawienie właściwości `gravity` układu LinearLayout na wartość `right` spowodujemy, że wszystkie widoki podrzędne zostaną przesunięte do prawej krawędzi układu:

```
android:gravity="right"
```

Właściwość `weight`

Właściwość `layout_weight` daje więcej miejsca na ekranie widokom posiadającym większą „wagę”. Znaczenie tej właściwości można najłatwiej zauważyć, tworząc układ `LinearLayout` w orientacji poziomej.

▼ Spróbuj sam!

Ustawianie wagi formantów w układach

Aby dowiedzieć się, jak działa atrybut `layout_weight` reprezentujący „wagę” widoku i jak należy używać go w swoich projektach, powinieneś wykonać polecenia przedstawione poniżej:

1. W folderze `/res/layout` utwórz nowy plik XML układu i nadaj mu nazwę `weight_example.xml`.
2. Użyj układu `LinearLayout` z orientacją poziomą.
3. Dodaj dwa przyciski do układu. Szerokość obu przycisków (atrybut `width`) powinna być ustawiona na stałą `match_parent`.
4. Dla obu przycisków ustaw atrybut `layout_weight` na wartość 1. Oba przyciski powinny mieć takie same rozmiary i zajmować całą szerokość kontenera.
5. Zmień wagę jednego z przycisków i sprawdź, jaki będzie to miało wpływ na wygląd przycisków w Android Studio.

Układ `FrameLayout`

Układ `FrameLayout` to kontener, który nie posiada żadnych specjalnych reguł rozmieszczania widoków. Domyślnie wszystkie widoki są rysowane w lewym górnym rogu ekranu. Takie rozwiązanie może się wydawać niezbyt użyteczne, ale w praktyce dzięki zastosowaniu odpowiednich ustawień marginesów pozwala na umieszczanie formantów w dowolnym miejscu. Układ `FrameLayout` jest przydatny w sytuacji, kiedy dwa formanty powinny być umieszczone w tym samym miejscu jeden na drugim. Przykładem może być sytuacja, gdy umieszczamy pole `TextView` na elemencie graficznym lub kiedy używamy kilku nałożonych na siebie obrazów do osiągnięcia efektu cienia i podświetlenia.

Dzięki zastosowaniu układu `FrameLayout` i odpowiednio dobranych atrybutów `layout_margins` możemy tworzyć bardzo efektowne i użyteczne interfejsy użytkownika.

Jeżeli korzystasz z układów `LinearLayout` lub `RelativeLayout`, to system Android podczas renderowania interfejsu użytkownika uwzględnia fizyczne rozmiary ekranu. Ponieważ w takich układach poszczególne widoki są rozmieszczone względem siebie bądź względem krawędzi kontenera, system ma większe możliwości optymalizacji rozłożenia elementów i zwiększenia funkcjonalności interfejsu. Z tego powodu używanie układów `LinearLayout` i `RelativeLayout` jest mocno zalecane. W praktyce w zdecydowanej większości przypadków najbardziej rekomendowanym rozwiązaniem będzie użycie układu `RelativeLayout`.

Podsumowanie

W tej godzinie dowiedziałeś się, czym są układy graficzne interfejsów użytkownika. Bardziej szczegółowo omówiliśmy układy `LinearLayout` i `RelativeLayout` ↪`Layout`, dzięki czemu miałeś możliwość bliżej zapoznać się z zasadami ich działania. Układy XML, pozwalające na definiowanie i rozmieszczanie widoków podrzędnych na ekranie, są wykorzystywane do tworzenia interfejsów użytkownika. Korzystając z atrybutów takich jak `padding`, `margins` i innych, masz pełną kontrolę nad wyglądem interfejsu użytkownika tworzonej aplikacji.

Pytania i odpowiedzi

P: Czym jest orientacja w układzie `LinearLayout`?

O: W układzie `LinearLayout` atrybut orientacji jest odpowiedzialny za kierunek rozmieszczania widoków. Orientacja pionowa powoduje, że poszczególne widoki są układane w kolumnie, podczas gdy orientacja pozioma tworzy wiersz. Atrybut orientacji nie jest wykorzystywany w układach `FrameLayout` ani `RelativeLayout`. Poszczególne układy mają różne reguły rozmieszczania widoków podrzędnych, co oznacza, że ich zestawy atrybutów również mogą się od siebie różnić.

P: Jaka jest najważniejsza różnica pomiędzy układami `FrameLayout` i `RelativeLayout`?

O: W układzie `RelativeLayout` poszczególne formanty są rozmieszczane względem siebie, czego nie możesz zrobić, korzystając z układu `FrameLayout`.

Warsztaty

Quiz

1. Czym różni się od siebie marginesy układu i `padding`?
2. Jaki jest związek między układem `LinearLayout` a klasą `ViewGroup`?
3. W jaki sposób możesz umieścić przycisk `Button` 100 pikseli od górnej krawędzi ekranu urządzenia? Skorzystaj z pikseli `dp` (piksele niezależne od gęstości wyświetlacza).

Odpowiedzi

1. Marginesy układu definiują dystans pomiędzy komponentem interfejsu a krawędzią układu. `Padding` ma bezpośredni wpływ na rozmiar komponentu. W zależności od rodzaju komponentu ustawienia `paddingu` mogą mieć różne efekty.
2. Układ `LinearLayout` jest rozszerzeniem klasy `ViewGroup`. Klasa `ViewGroup` w systemie Android to kontener, który zawiera inne widoki.

3. Aby ustawić górny margines na 100 pikseli dp, powinieneś użyć następującego ustawienia:

```
android:layout_marginTop="100 dp"
```

Ćwiczenia

1. Korzystając z układu `FrameLayout`, zaimplementuj interfejs użytkownika z przyciskami `Button` znajdującymi się w każdym z narożników ekranu. Spróbuj zrobić to samo z użyciem układu `RelativeLayout`.
Aby to zrobić z użyciem układu `FrameLayout`, powinieneś wykonać polecenia przedstawione poniżej:
 - ▶ Utwórz plik układu `FrameLayout`.
 - ▶ Utwórz cztery przyciski `Button`.
 - ▶ Ustaw atrybuty `topMargin` i `leftMargin` poszczególnych przycisków `Button` tak, aby rozmieścić je w narożnikach ekranu.
2. Spróbuj umieścić jeden układ wewnątrz drugiego. Aby to zrobić, utwórz układ `LinearLayout` z orientacją pionową. Dodaj do niego układ `FrameLayout` i przycisk `Button`, a następnie spróbuj dodać formant `ImageView` do układu `FrameLayout`.
3. Utwórz układ `RelativeLayout`, dodaj do niego formanty `ImageView` oraz `TextView`, a następnie dołącz ten układ do innego układu za pomocą znacznika `include`.

Skorowidz

A

- adapter, 155
 - ArrayAdapter, 156, 159, 200
 - BaseAdapter, 172, 215
 - CursorAdapter, 367, 371, 378
 - PieDbAdapter, 354
 - RecyclerView.Adapter, 190
 - SimpleCursorAdapter, 316
- adres URL, 341
- akcelerometr, 419
- akcja
 - ACTION_DIAL, 56
 - ACTION_PICK, 56
 - ACTION_SEND, 56
 - ACTION_VIEW, 53
- aktualizacja
 - danych, 308
 - definicji usługi, 113
 - interfejsu użytkownika, 110
 - manifestu, 358
 - narzędzi, 47
 - powiadomień, 388
 - systemu, 418
- aktywne połączenie sieciowe, 334
- aktywności, 26, 81
 - główne, 37
 - metody zwrotne, 96
 - niszczenie, 94
 - puste, 42, 99
 - tworzenie, 94, 98
 - uruchamianie, 94
 - wznawianie, 94
 - z Google Play, 410
 - z kontenerem, 157
 - z menu wysuwany, 268
 - z RecyclerView, 187
 - zatrzymywanie, 94
 - zawieszanie, 94
- aktywność
 - MainActivity, 49
 - SecondaryActivity, 49
 - ActionBarBasicActivity, 255
 - ChildActivity, 272

- DatabaseActivity, 312
- MainActivity, 92
- MainActivity.java, 117
- MessageActivity, 83–86, 94
- PhotoActivity, 371
- PreferenceActivity, 299
- alternatywne zasoby aplikacji, 74
- Android
 - Beam, 422
 - Lollipop, 24, 427
 - SDK, 74
 - Studio, 24, 32, 120
 - TV, 395, 403
 - Virtual Device Manager, 46
 - Wear, 395, 396, 400
- animacje, 180, 196
- API, application programming interface, 320
- API serwisu Flickr, 338
- aplikacja
 - Google Maps, 53
 - Settings, 76
- aplikacje
 - bezpłatne, 435
 - dla Android TV, 403–406
 - dla Android Wear, 396
 - płatne, 435
 - z płacnościami wewnętrznymi, 435
 - z reklamami, 435
- asynchroniczne zadania, 109
- atrybut, 149
 - dependency, 295
 - layout_weight, 152
 - showAsAction, 251
- atrybuty ScaleType, 218

B

- baza danych SQLite, 275, 303
- biblioteka, 184
 - Gson, 328
 - leanback, 406
 - Picasso, 230, 417
- bitmapy, 217
- Bluetooth, 422

C

certyfikat, 432
 chmura, 319
 cienie, 178
 cykl życia aktywności, 93
 czcionka Roboto, 180
 czujnik

- natężenia światła, 419
- orientacji, 419
- przyspieszeń, 419
- zbliżeniowy, 419

D

dane z kalendarza, 343, 350
 definicja
 definiowanie

- dedykowanych zasobów językowych, 77
- tablicy, 159, 306
- usługi, 113
- zapytania, 310
- zasobów domyślnych, 77

 deklarowanie dostawcy treści, 356
 dodawanie

- animacji, 196
- aplikacji do sklepu, 433
- bibliotek, 184
- danych, 159, 200, 308
- elementów nawigacji, 255
- formantów EditText, 131
- formantów TextView, 127
- ikon, 253, 388
- obiektu MediaController, 237
- obrazu do przycisku, 133
- powiadomienia, 390
- przycisku, 41, 54
- pustej aktywności, 34
- widoku podrzędnego, 260

 dokumentacja online, 30
 dostawca treści, 341, 353

- MyContentProvider, 362

 dostęp do chmury, 319
 dostępność usług, 412
 dostosowywanie

- powiadomień, 390
- stylów, 420

 dpi, dots per inch, 70
 działanie

- adapterów, 160
- aktywności, 82
- dostawcy treści, 354
- loaderów, 367
- w tle, 105
- w wątku UI, 107

E

edytor, 35
 eksploracja projektu, 34
 eksportowanie pliku certyfikatu, 432
 elementy kalendarza, 347
 elewacja, 182

- formantów, 183

 emulator, 46
 external storage, 277

F

FAB, Floating Action Button, 178, 192
 formant, 199

- AutoCompleteTextView, 202
- Button, 86
- CheckBox, 202
- EditText, 87, 126, 131, 138
- ListView, 155
- ProgressBars, 204
- RadioButton, 202, 204
- SeekBar, 205
- Spinner, 199, 200, 259, 261
- TextView, 48, 73, 86, 126, 129
- VideoView, 148

 formanty

- Android Studio, 125
- interfejsu użytkownika, 125
- układów, 150

 format

- JSON, 320, 325
- MP3, 243
- XML, 337

 formatowanie zasobów, 66
 formaty kolorów, 68
 fragment, 81, 97

- PhotoListFragment, 332

 funkcje API, 321

G

generowanie aktywności, 299
 gesty, 420
 Google

- Inbox, 175
- Maps, 53
- Now, 181
- Play, 409
- Play Services, 411

 GPS, Global Positioning System, 412
 grafika, 178
 grupa Square, 417
 grupy widoków, 421

I

IDE, Integrated Development Environment, 24
 identyfikator
 _id, 306
 URI, 52, 341, 353
 zasobu, resource ID, 82
 ikony, 428
 implementacja
 menu wysuwanego, 264
 metody
 BindView(), 376
 delete(), 361
 getType(), 360
 insert(), 361
 newView(), 375
 onCreateLoader(), 373
 onLoaderReset(), 374
 onLoadFinished(), 373
 query(), 358
 update(), 361
 śledzenia położenia, 414
 importowanie klas, 45
 informacje
 dodatkowe, 27
 o aplikacji, 434
 o kalendarzu, 343
 o właścicielu, 377
 o właściwości, 134
 inicjowanie loadera, 369
 instalowanie
 biblioteki Picasso, 231
 pakietu Android Studio, 32
 instrukcja switch, 267
 intencja INSERT, 351
 intencje, 26, 39, 102, 235, 246
 jawne, 50
 niejawne, implicit Intent, 27, 50
 interfejs
 API, 276, 320, 338
 API kalendarza, 343
 LoaderCallbacks, 371
 Material Design, 175, 177
 NFC API, 422
 OpenGL ES API, 422
 WebViewClient, 213
 interfejsy
 płaskie, 176
 użytkownika, 40, 110, 123, 141
 internal storage, 276

J

język XML, 144
 JSON, JavaScript Object Notation, 325

K

kalendarz, 342
 dane, 343, 350
 kolumna danych, 352
 pobieranie danych, 344
 wyświetlanie danych, 346
 wyświetlanie zdarzeń, 348
 kamera, 421
 kanał alfa, 223
 karty, 181
 karty preferencji, 297
 katalog
 values, 63
 values-820dp, 64
 DIRECTORY_PODCASTS, 286
 drawable, 63
 drawable-hdpi, 71
 drawable-mdpi, 71
 drawable-xhdpi, 71
 extra-high density, 71
 high density, 71
 res, 61, 63
 values, 63, 73
 katalogi publiczne, 283
 klasa
 ActionBarActivity, 264
 Activity, 26, 30, 32
 android.provider.CalendarContract, 343
 Application, 31
 AsyncTask, 109, 111
 BaseAdapter, 160, 161
 Bitmap, 224
 BitmapFactory.Options, 226, 232
 BroadcastReceiver, 31
 BroadcastReceivers, 28
 Canvas, 224, 229
 ContentProvider, 354
 ContentResolver, 345, 365
 Cursor, 312
 CursorAdapter, 374
 CursorLoader, 368
 DatabaseHelper, 305
 DelayReceiver, 115
 FileOutputStream, 283
 FrameLayout, 263
 GooglePlayServicesUtil, 412
 HelpActivity, 31
 URLConnection, 323, 324
 Intent, 27, 30
 IntentService, 31, 112
 IntentServices, 27
 JSONArray, 327
 JSONObject, 326
 JsonReader, 326

klasa

- Loader, 368
- LoaderManager, 369
- LoaderManagerCallbacks, 369
- LoadPhotos, 335
- Matrix, 223
- MediaPlayer, 242
- MyContentProvider, 356
- Notification, 384
- Notification.Builder, 384
- NotificationManager, 384
- PendingIntent, 384
- PhotoCursorAdapter, 375
- PhotoListFragment, 333
- Pie, 314
- PieAdapter, 161
- PieDbAdapter, 305–310, 313
- PlaceholderFragment, 101
- PreferencesFragment, 292
- RelativeLayout, 146, 263
- Service, 112
- SharedPreferences, 275, 287–291
- SimpleCursorAdapter, 314
- SQLiteOpenHelper, 305
- StringBuilder, 331
- View, 127, 167
- ViewGroup, 259
- ViewHolder, 190
- WatchViewStub, 399

klasy loaderów, 368

klauzula

- GROUP BY, 311
- ORDER BY, 317
- where, 310, 311

klawiatura ekranowa, 131

klucz

- cyfrowy, 429
- interfejsu API, 321
- MESSAGE_DATA, 87, 88

kod

- aktywności DatabaseActivity, 312
- aktywności z menu wysuwanym, 268
- dostawcy treści, 355
- klasy PhotoListFragment, 333
- uruchamiający aktywność, 44
- źródłowy MainActivity.java, 117

kolor, 68, 179

komponent, 409

- FrameLayout, 263
- ListFragment, 319
- RelativeLayout, 263
- RippleDrawable, 195

komponenty powiadomień, 384

komunikacja peer-to-peer, 422

komunikat ANR, 106

konfigurowanie

- menu wysuwanego, 264
- projektu aplikacji, 397
- urządzenia, 90
- formantu AutoCompleteTextView, 202
- usług Google Play, 410

konstruktor klasy, 190

kontener

- AdapterViewFlipper, 210
- CardView, 183, 185, 191
- GridView, 209
- ListView, 155–160, 186
- RecyclerView, 183, 186, 192
- ScrollView, 215

kontroler MediaController, 237

korzystanie

- z baz danych, 312
- z kontenera ScrollView, 215
- z palety formantów, 125
- z piaskownicy, 322
- z systemu plików, 275
- z zasobów układu, 144

kursor, 310, 312

kwalifikatory katalogów, 74

L

listener

- onClickListener, 86
- onConnectionFailed(), 411

loader CursorLoader, 371

loadery, 367

- klasy, 368
- resetowanie, 371
- stany, 369
- tworzenie, 370

lokalizacja urządzenia, 51, 412, 414, 416

M

magnetometr, 419

manifest aplikacji, 26

marginesy układów, 149, 153

maszyna wirtualna, 401

menu, 253

- opcji, 249
- przepełnienia, overflow menu, 250
- wysuwane, 264, 266, 268

metoda

- append(), 331
- bindView(), 376
- changeImage(), 136
- closeDrawer(), 267
- Context.getCacheDir(), 279
- Context.getExternalFilesDir(), 279

- Context.getFilesDir(), 278, 283
 - delete(), 362
 - doInBackground(), 109, 331, 332
 - Environment.getExternalStoragePublicDirectory(), 279
 - findViewById(), 44, 169
 - flickr.photos.getRecent(), 321
 - getApplicationContext(), 32
 - getAssets(), 244
 - getContentResolver(), 365
 - getContentResolver.query(), 364
 - getCount(), 161
 - getExternalFilesDir(), 283, 284
 - getExternalFilesStoragePublicDirectory(), 283, 285
 - getFilesDir(), 277
 - getIntExtra(), 31
 - getItem(), 161
 - getItemCount(), 190
 - getItemId(), 161, 254
 - getPhotos(), 330
 - getPieFromCursor(), 312
 - getResources(), 138
 - getResources.getDrawable(), 218
 - getSharedPreferences(), 288
 - getStringExtra(), 87
 - getSupportFragmentManager(), 100
 - getText(), 137
 - getType(), 360
 - getView(), 161, 169, 173
 - getType(), 357
 - initLoader(), 373
 - insert(), 361
 - Intent.resolveActivity(), 51
 - makePies(), 163, 169
 - newView(), 375
 - onActivityResult(), 87
 - onBindViewHolder(), 190
 - onClick(), 44
 - OnClickListener, 44
 - onClickListener(), 53
 - onCreate(), 35, 82, 96, 99, 102, 189, 307
 - onCreateLoader(), 373
 - onCreateMenuOptions(), 253
 - onCreateOptionsMenu(), 250
 - onCreateView(), 100
 - onCreateViewHolder(), 190
 - onDestroy(), 96, 97
 - onHandleIntent(), 31, 115
 - onItemClick(), 158
 - OnItemClickListener(), 190
 - onLoaderReset(), 374
 - onLoadFinished(), 370, 373, 378
 - onLocationChanged(), 424
 - onOptionsItemSelected(), 253
 - onPause(), 96, 97
 - onPostExecute(), 109, 111, 332
 - onPreExecute(), 109
 - OnPrepared(), 242
 - OnPreparedListener, 242
 - onPreparedListener(), 246
 - onProgressUpdate(), 109, 205
 - onReceive(), 31
 - onResume(), 96, 97
 - onSaveInstanceState(), 91
 - onStart(), 95, 96
 - onStop(), 96, 97
 - openFileOutput(), 283
 - post(), 108
 - preExecute(), 110
 - publishProgress(), 109, 205, 208
 - putExtra(), 31, 88
 - query(), 358, 359
 - registerOnSharedPreferenceChangeListener(), 293
 - RemoteViews.setOnClickPendingIntent(), 393
 - saveInstanceState.putString(), 91
 - seekTo(), 247
 - sendBroadcast(), 28, 31, 115
 - setContentView(), 82, 93, 95, 144
 - setImageDrawable(), 137
 - setOnClickListener(), 44, 107, 138
 - setTag(), 173
 - setText(), 137
 - setVisibility(), 207
 - showList(), 332
 - startActivity(), 27, 45
 - startActivityForResult, 82, 88
 - startActivityForResult(), 103
 - startService(), 27
 - swapCursor(), 370
 - SystemClock.sleep(), 106
 - update(), 361
 - View.post(), 108
 - metody
 - adaptera bazy danych, 305
 - klasy SharedPreferences, 291
 - tabela, 305
 - zwrotne, 95, 96
 - miniatura obrazu, 233
 - modyfikowanie zapytań, 374
 - motyw aplikacji, 182
 - motywy, 420
- ## N
- narzędzie
 - Flickr API, 322
 - jarsigner, 431
 - keytool, 431

nasłuchiwanie stanów widoku, 240
 nawigacja, 255
 up navigation, 256
 nawigowanie w aplikacji, 249
 nazwa pakietu, 42
 NFC, Near Field Communication, 422
 nieoznaczony pasek postępu, 204

O

obiekt

 Bundle, 92
 FlickrPhoto, 328
 FragmentManager, 100
 InputStream, 324
 JSONObject, 328, 329
 MediaController, 237
 MediaPlayer, 242
 Pie, 169, 190, 304
 Service, 112
 View, 173
 View.OnClickListener, 44
 ViewHolder, 168, 190

obiekty klasy

 Activity, 26
 BroadcastReceiver, 26
 BroadcastReceivers, 28
 Intent, 26, 27
 IntentService, 26
 IntentServices, 27

obracanie obrazów, 222

obrazy o dużych rozmiarach, 225, 226

obsługa

 danych, 127
 gestów, 420
 intencji niejawnych, 58
 kontenera ListView, 266
 multimediów, 245
 naciśnięcia przycisku, 44, 134
 niejawnej intencji, 60
 ustawień regionalnych, 76
 zdarzeń, 239
 zmian w konfiguracji urządzenia, 90, 92

odbieranie

 odpowiedzi, 324
 przekazywanej intencji, 48

odbiornik rozgłoszeń BroadcastReceiver, 115

odbiorniki

 komunikatów, 28
 rozgłoszeń, 26

odczytywanie

 danych, 289, 314
 aplikacji, 280
 w preferencjach, 290
 z kursora, 305

 preferencji, 289, 299
 tablicy, 327
 odtwarzanie plików
 audio, 243
 wideo, 235, 239
 OOM, out-of-memory, 225
 opcja
 Blank Activity, 33
 Phone and Tablet, 33
 uses-screens, 427
 Wear, 33

opcje

 powiadomień, 389
 preferencji ListPreference, 296

OpenGL ES, 422

oprogramowanie open source, 417

orientacja w układzie, 153

OS, Operating System, 23

P

padding, 150

pakiet

 Android SDK, 74, 417
 Android Studio, 32
 android.gesture, 420
 Crashlytics, 418
 Fabric, 418
 Realm, 418

paleta

 formantów, 125
 kolorów, 179

pamięć masowa

 wewnętrzna, 275
 zewnętrzna, 275

pamięć podręczna, 282

panel

 Component Tree, 41
 Properties, 41

para klucz-wartość, 287, 288

parametr DATABASE_VERSION, 307

parametry zapytania, 311

parsowanie danych, 325, 328

pasek

 ActionBar, 249, 255–257, 272
 postępu ProgressBar, 111
 ProgressBar, 207, 216
 SeekBar, 207, 224
 Toolbar, 256–261, 272

piaskownica serwisu Flickr, 322

piksele logiczne, 70

platforma mobilna, 25

plik

 activity_main.xml, 40, 63, 68, 83, 99, 156
 AndroidManifest.xml, 26, 35, 92

- background_shape.xml, 71
- build.gradle, 186, 406
- certyfikatu, 432
- color.xml, 68
- colors.xml, 195
- DelayService.java, 113
- dimens.xml, 64, 69
- fab_anim.xml, 195
- fab_background.xml, 195
- fab_icons.xml, 195
- fab_layout.xml, 195
- FloatingActionButton.java, 195
- fragment_main.xml, 99, 101
- ic_launcher.png, 133
- MainActivity.java, 40, 44, 49, 85, 99, 101
- manifestu aplikacji, 426
- menu_main.xml, 63, 250
- MessageActivity.java, 89
- pie_view_item.xml, 164
- preferences.xml, 292
- SecondaryActivity.java, 48, 58
- strings.xml, 63, 136
- styles.xml, 63, 72
- zasobów menu, 253
- pliki
 - audio, 243, 284
 - graficzne, 284
 - preferencji, 287, 289
 - tymczasowe aplikacji, 278
 - układu, layout files, 61, 65, 84, 141
 - wideo, 235, 284
 - XML, 79
- płótno, 224, 229
- pobieranie
 - danych, 310, 323, 353, 362
 - danych w tle, 330
 - danych z kalendarza, 344
 - informacji, 199
 - obiektów, 312
 - zdalnych danych, 320
- podkasty, 284
- podpis cyfrowy pakietu, 430
- podpisywanie aplikacji, 428
- pole
 - EditText, 199
 - KEY_ROWID, 306
- polecenie
 - jarsigner, 432
 - keytool, 431
- połączenia
 - Bluetooth, 422
 - sieciowe, 334
- postęp realizacji zadania, 204
- powiadomienia, 383
 - aktualizowanie, 388
 - dostosowywanie, 390

- klasy, 384
- komponenty, 384
- opcje, 389
- tworzenie, 385, 401
- z powiązaną akcją, 386
- zarządzanie, 385
- pozycjonowanie
 - formantu, 128
 - odtworzenia wideo, 239
- preferencja, 289
 - CheckBoxPreference, 293
 - EditTextPreference, 294, 296
 - ListPreference, 294, 295
 - MultiSelectListPreference, 294
 - SwitchPreference, 295
- projektowanie
 - aplikacji, 175
 - układów, 142
 - widoków, 421
- protokół HTTP, 320
- prywatne dane aplikacji, 278, 279
- przechowywanie
 - danych, 287
 - plików, 276, 277
- przeglądanie zasobów aplikacji, 64
- przekazywanie danych, 47
- przełączanie stron, 211
- przestrzeń 3D, 178
- przewijanie w poziomie, 211
- przezroczystość, 223
- przycisk, 41
 - App Theme, 182
 - Button, 87, 132
 - ImageButton, 133
 - Odtwarzanie, 239
 - Pauza, 239
 - Pokaż mapę, 54
- przyciski
 - FAB, 192
 - pływające, 178, 192
- publikowanie aplikacji, 425, 433

R

- resetowanie loadera, 371
- responsywność, 105
- rodzaje układów, 150
- rozmiar tekstu, 131
- rozszerzanie klasy, 292
 - BaseAdapter, 160, 161
- rysowanie na płótnie, 229

S

SDK, Software Development Kit, 33, 74, 417
 sensory, 419
 serwis Flickr, 326
 siatka, 209
 skalowanie obrazu, 219
 skeuomorfizm, 176
 sklep
 Amazon, 434
 Google Play, 433
 sprawdzanie
 dostępności usług, 411
 połączenia sieciowego, 334
 stała
 CURSOR_ITEM_BASE_TYPE, 365
 MESSAGE_REQUEST_CODE, 86
 stany
 loadera, 369
 widoku, 207
 widoku VideoView, 240
 strona WWW, 55
 struktura
 aplikacji, 325, 330
 tablicy, 327
 styl, 72, 420
 CustomTextStyle, 73
 tekstu, 131
 symbol @, 132
 system
 GPS, 413
 operacyjny, OS, 23
 plików, 275

Ś

śledzenie położenia, 414
 środowisko programistyczne, 24

T

tabela, 303, 306
 INSTANCES, 352
 tablica
 ArrayList, 169, 190, 329
 JSONArray, 327
 technologia
 Beam, 422
 NFC, 422
 termometr, 419
 testowanie aplikacji, 400, 406
 tworzenie
 adapterów, 374
 aktywności, 98

aplikacji, 26, 330
 baz danych, 303
 dostawców treści, 353–356, 364
 drugiej aktywności, 42
 fragmentów, 98
 fragmentów PreferencesFragment, 292
 ikon, 428
 intencji, 54
 interfejsów użytkownika, 40, 123
 klucza, 431
 klucza interfejsu API, 321
 kontenera CardView, 183, 185
 loadera, 370
 maszyny wirtualnej, 401
 nowego klucza, 430
 nowego projektu, 33
 nowej aktywności, 43
 obiektów, 326
 obiektu FlickrPhoto, 328
 odbiornika rozgłoszeń, 115
 paska ActionBar, 255
 paska Toolbar, 258
 pliku zasobów, 67
 powiadomień, 385, 401
 rozszerzenia klasy, 161
 tablicy ArrayList, 329
 układów, 144
 układów interfejsów użytkownika, 141
 układu dla powiadomienia, 390
 URI, 353
 usługi IntentService, 120
 zapytania, 316
 żądania, 324
 typografia, 179
 typy
 danych, 291
 preferencji, 294, 301

U

udostępnianie strony internetowej, 29
 układ
 FrameLayout, 151, 152
 GridLayout, 151
 LinearLayout, 150, 151, 197
 RelativeLayout, 142–145, 148, 150, 266
 TableLayout, 151
 układy
 dla powiadomienia, 390
 do wyświetlania danych, 164
 do wyświetlania fragmentów, 101
 interfejsów użytkownika, 141
 uprawnienie
 READ_EXTERNAL_STORAGE, 277
 WRITE_EXTERNAL_STORAGE, 277

URI, Uniform Resource Identifiers, 341
 URL, Uniform Resource Locator, 341
 uruchamianie
 akcji, 132
 aktywności, 39, 59
 aplikacji, 46, 47
 usługi IntentService, 114
 urządzenia
 Android TV, 403
 Android Wear, 396, 400
 wearable, 395
 usługa, 26, 27, 37
 DelayService, 114, 115
 IntentService, 112, 113, 114, 120
 Service, 112
 usługi
 Google Play, 409
 lokalizacyjne, 412
 ustawienia regionalne, 76
 usuwanie danych, 308
 używanie
 powiadomień, 383
 sensorów, 419

W

wartości atrybutu showAsAction, 251
 wartość null, 311
 wątek
 Thread, 108
 UI, 107
 UI Thread, 27, 105
 wersje językowe, 76
 wewnętrzny obszar przechowywania plików,
 276
 widok, 127, 199
 AdapterViewFlipper, 212
 Gallery, 212
 HorizontalScrollView, 212
 ImageView, 217
 Project, 62
 VideoView, 235, 236
 ViewFlipper, 212
 ViewPager, 212
 WebView, 212, 214
 widoki podrzędne, 163, 165, 259
 właściwości przycisków Button, 132
 właściwość
 android:checked, 203
 hint, 138
 inputType, 131, 132
 onClick, 136, 138
 padding, 149
 ScaleType, 220, 221
 textSize, 159
 visibility, 207
 wprowadzanie danych, 127
 współrzędne geograficzne, 52
 wybieranie
 elementu kalendarza, 347
 motywu aplikacji, 182, 183
 urządzenia, 399
 zasobu koloru, 69
 wygląd aplikacji, 62
 wyjątek OOM, 225
 wykorzystywanie zdarzeń widoku, 240
 wyrównywanie
 do krawędzi, 145
 formantu, 146
 widoków względem siebie, 146
 wysuwane panele menu, 261
 wyświetlanie
 danych, 158, 164, 208, 314
 danych w siatce, 209
 danych z kalendarza, 346
 dużych obrazów, 226
 fragmentów, 101
 kart preferencji, 297
 klawiatury ekranowej, 131
 kodu, 35
 kontenerów RecyclerView, 186
 listy, 332
 lokalizacji, 51, 54, 55
 menu opcji, 249
 obrazów, 217
 paska postępu, 111, 205, 206
 powiadomień, 383, 386
 strony WWW, 55
 widoku, 214
 zdarzeń z kalendarza, 348
 zdjęć, 368
 wywoływanie
 aktywności, 44, 46
 funkcji API, 321
 wzorzec ViewHolder, 167

Z

zadanie
 AsyncTask, 109, 110, 325, 330
 LoadPhotos, 331
 zapisywanie
 danych aplikacji, 280, 282, 285
 danych w preferencjach, 290
 preferencji, 289, 291
 prywatnych danych, 279
 zapytania, 305, 310
 zarządzanie
 danymi, 305
 powiadomieniami, 385

- zasoby aplikacji, 61
 - alternatywne, 74
 - korzystanie, 65
 - kwalifikatory katalogów, 74
 - proste zasoby, 66
 - tworzenie pliku, 67
 - typu color, 67
 - typu dimensions, 69
 - typu drawable, 71
 - typu String, 129, 135
 - wersje językowe, 76
- zastosowanie
 - adaptera PieDbAdapter, 354
 - adapterów ArrayAdapter, 156
 - biblioteki Picasso, 230
 - elementów RippleDrawable, 195
 - formantu AutoCompleteTextView, 202
 - formantu Spinner, 199, 200
 - intencji, 235, 236
 - intencji niejawnych, 51
 - klasy BitmapFactory.Options, 226
 - klasy JSONArray, 327
 - klasy Matrix, 223
 - klasy PieDbAdapter, 313
 - klasy SharedPreferences, 287
 - klasy SimpleCursorAdapter, 314
 - kontenera AdapterViewFlipper, 210
 - kontenera GridView, 209
 - kontenerów RecyclerView, 185
 - paddingu, 150
 - paska ActionBar, 249
 - paska SeekBar, 224
 - paska Toolbar, 256
 - plików układu, 82
 - preferencji, 297
 - stylów, 72
 - układu LinearLayout, 151
 - usługi IntentService, 112
 - widoku VideoView, 236, 238
 - wysuwanych paneli menu, 261
 - wzorca ViewHolder, 167, 168
 - zadań asynchronicznych, 109
- zdalny interfejs API, 319
- zdarzenia, 239
 - kalendarza, 348
 - widoku VideoView, 240
- zewnątrzny obszar przechowywania plików, 277
- zintegrowane środowisko programistyczne, IDE, 24
- zmiana
 - koloru tekstu, 130
 - nazwy przycisku, 42
 - orientacji urządzenia, 90
 - wyświetlanego tekstu, 129
- znaczenie układów, 262
- znacznik include, 142
- zwracanie wyników działania aktywności, 82

Ż

żądanie, 324

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>

Perspektywy rozwoju Androida są wyjątkowo obiecujące. System ten przebojem zdobywa nowe obszary rynku, a jego zalety, takie jak modułowa architektura, wysoka elastyczność czy otwarty charakter, są cenione zarówno przez programistów, jak i użytkowników aplikacji. Rocznie sprzedaje się miliardy urządzeń pracujących pod kontrolą Androida, a jeśli wziąć pod uwagę rozwój technologii i mnożące się pomysły na wyposażenie w procesor kolejnego przedmiotu codziennego użytku, można przepowiadać temu systemowi świetlaną przyszłość. Podobnie zresztą jak programistom, którzy nauczą się efektywnie pisać świetne, atrakcyjne aplikacje dla Androida.

Książka, którą trzymasz w dłoni, została pomyślana jako podręcznik. Nawet jeśli jesteś bardzo początkującym programistą, błyskawicznie — bo po lekturze 24 rozdziałów, z których każdy zajmie najwyżej godzinę — poznasz podstawy programowania dla Androida i szybko zaczniesz tworzyć w pełni funkcjonalne aplikacje. Najpierw dowiesz się, czym są aktywności, intencje, usługi i odbiorniki rozgłoszeń. Później zagłębisz się w bardziej zaawansowane zagadnienia: nauczysz się obsługi multimediów, poznasz interfejs Material Design firmy Google, sprawdzisz, jak wykorzystywać bazy danych SQLite. Wisienką na tym torcie atrakcji będzie pokazanie szczególnych możliwości systemu Android, wykorzystywanych w bardzo profesjonalnych aplikacjach.

Szczególnie ciekawe dla Ciebie będą:

- omówienie najnowszych możliwości i mechanizmów systemu Android 5 (Lollipop), pozwalających na tworzenie aplikacji dla Androida TV i urządzeń typu Android Wear
- przedstawienie świetnego narzędzia dla programistów — środowiska Android Studio
- nauka projektowania responsywnych aplikacji, wykorzystujących zadania działające w tle
- wskazówki dotyczące tworzenia wyrafinowanych systemów nawigowania w aplikacji, korzystania z pasków ActionBar oraz z wysuwanych menu
- sposoby pobierania danych z chmury i parsowania danych zapisanych w formacie JSON
- użycie Google Play Services do lokalizowania położenia urządzenia

Carmen Delessio — doświadczony deweloper aplikacji. Pracował jako programista, architekt rozwiązań i CTO w dużych i małych firmach. Jest autorem wielu nagradzanych aplikacji dla systemu Android.

Lauren Darcey — uznany autorytet w dziedzinie architektury aplikacji i projektowania komercyjnych aplikacji mobilnych.

Shane Conder — twórca wielu popularnych aplikacji dla systemów Android, iOS, BREW, BlackBerry, J2ME, Palm i Windows Mobile.



Kreatywnych ogranicza tylko wyobraźnia!

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

Helion

SAMS

42134 numer katalogowy

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

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

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

ISBN 978-83-283-2150-2



9 788328 321502

Informatyka w najlepszym wydaniu

cena: 79,00 zł