

Shane Conder, Lauren Darcey

WYCZERPUJĄCY PRZEWODNIK,
GRUNTOWNIE ZAKTUALIZOWANY
W OPARCIU O NAJNOWSZĄ WERSJĘ ANDROID SDK
I NAJLEPSZE TECHNIKI PROGRAMOWANIA!

Android

WYDANIE II

**PROGRAMOWANIE APLIKACJI
NA URZĄDZENIA PRZENOŚNE**



» Idź do

- Spis treści
- Przykładowy rozdział
- Skorowidz

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
© Helion 1991–2011

Android. Programowanie aplikacji na urządzenia przenośne. Wydanie II

Autorzy: [CShane Conder](#), [Lauren Darcey](#)

Tłumaczenie: Piotr Rajca

ISBN: 978-83-246-3349-4

Tytuł oryginału: [Android Wireless Application Development \(2nd Edition\)](#)

Format: 170×230, stron: 820



Wyczerpujący przewodnik, gruntownie zaktualizowany w oparciu o najnowszą wersję Android SDK i najlepsze techniki programowania!

Gdy w 2008 roku Google opublikował Androida, rynkiem telefonów komórkowych wprawdzie nieco zatrzęsło, ale nikt nie spodziewał się, że ta platforma aż tak poważnie zagrozi takim gigantom mobilnych systemów operacyjnych, jak iOS Apple, Windows Mobile, Symbian czy RIM BlackBerry. Od tego czasu każde nowe urządzenie z Androidem miało coraz większe możliwości i było jeszcze bardziej ekscytujące od swoich poprzedników. Minęły zaledwie trzy lata od dnia, kiedy na rynku pojawił się pierwszy telefon z systemem Android - T-Mobile G1, stworzony przez firmę HTC - a system ten już okrzyknięty został najszybciej sprzedającą się platformą dla telefonów przenośnych. To oczywiście nie pozostaje bez echa: potrzeba coraz więcej ludzi specjalizujących się w programowaniu aplikacji na tę fascynującą, darmową i otwartą platformę mobilną.

Oto kompletny podręcznik, zawierający wszystko, co potrzebne do tworzenia, wdrażania i sprzedawania aplikacji na urządzenia przenośne działające pod kontrolą najnowszych wersji Androida. Autorzy – w oparciu o swoje wieloletnie doświadczenie w tworzeniu mobilnych aplikacji – wprowadzą Cię we wszystkie etapy tego procesu: pomysł, projektowanie, pisanie kodu, testowanie, pakowanie i rozpowszechnianie aplikacji. Poznasz doskonale specyfikację platformy Android, podstawowe zasady efektywnego projektowania aplikacji na nią przeznaczonych oraz najlepsze praktyki związane z tworzeniem wygodnych interfejsów użytkownika. Znajdziesz tu także wyczerpujące opisy wszystkich kluczowych interfejsów programistycznych: do obsługi składowania danych, komunikacji sieciowej, obsługi rozmów telefonicznych, usług lokalizacyjnych, multimediów, grafiki 3D oraz opcjonalnych komponentów sprzętowych. Oprócz tego książka została uzupełniona praktycznymi sztuczkami, które pozwolą Ci zaoszczędzić sporo cennego czasu i uniknąć wielu niepotrzebnych pułapek!

Ponadto znajdziesz tu:

- kilka rozdziałów opisujących interfejs API do obsługi technologii związanych z WWW, a także Android NDK, poszerzanie zasięgu aplikacji, zarządzanie użytkownikami, synchronizację danych, tworzenie kopii bezpieczeństwa, zaawansowane metody wprowadzania danych
- wyczerpujące informacje na temat plików manifestu, dostawców treści, projektowania i testowania aplikacji
- prezentację najbardziej aktualnych i interesujących zagadnień, takich jak obsługa komunikacji Bluetooth i gestów, rozpoznawanie mowy, widgety, technologie Live Folders, Live Wallpapers oraz globalne wyszukiwanie
- aktualne informacje na temat generowania grafiki 3D przy użyciu OpenGL ES 2.0
- zagadnienia związane z zapewnianiem zgodności pomiędzy różnymi urządzeniami

Spis treści

O autorach	23
Wprowadzenie	25
Kto powinien przeczytać tę książkę	25
Kluczowe pytania, na jakie odpowiada ta książka	26
Struktura książki	27
Opis zmian wprowadzonych w tym wydaniu książki	28
Stosowane środowisko programistyczne	30
Dostępne materiały dodatkowe	30
Gdzie szukać dodatkowych informacji	30
Konwencje stosowane w książce	31
Kontakt z autorami	32
Część I	
Ogólne informacje o platformie Android	33
Rozdział 1	
Prezentacja systemu Android	35
Krótką historią tworzenia aplikacji dla urządzeń przenośnych	35
Dawno, dawno temu	35
„Cegła”	38
Wireless Application Protocol (WAP)	39
Zastrzeżone platformy mobilne	42
Open Handset Alliance	44
Google staje się bezprzewodowy	44
Tworzenie Open Handset Alliance	45
Producenci — projektowanie telefonów dla Androida	46
Operatorzy — dostarczanie wrażeń	47
Dostawcy treści — tworzenie aplikacji na Androida	47
Wykorzystanie wszystkich możliwości Androida	48
Co wyróżnia platformę Android?	48
Android — platforma następnej generacji	49
Darmowy i otwarty	50
Znane i niedrogie narzędzia programistyczne	51

Rozsądny stopień trudności nauki programowania	51
Umożliwianie tworzenia potężnych aplikacji	51
Bogate i bezpieczne możliwości integracji aplikacji	52
Brak kosztownych przeszkód utrudniających publikację	52
„Otwarty rynek” aplikacji	53
Nowa, rozwijająca się platforma	54
Platforma Android	54
Architektura Androida	55
Bezpieczeństwo i uprawnienia	56
Pisanie aplikacji na Androida	58
Podsumowanie	60
Odwołania i inne źródła informacji	60
Rozdział 2 Przygotowywanie środowiska programistycznego	61
Konfiguracja środowiska programistycznego	61
Konfiguracja własnego systemu do debugowania urządzenia	62
Konfiguracja urządzenia do testowania	63
Aktualizacja Android SDK	64
Problemy z Android SDK	64
Poznajemy Android SDK	65
Zrozumienie postanowień licencji	65
Korzystanie z dokumentacji Android SDK	66
Prezentacja szkieletu programowania aplikacji	68
Poznanie narzędzi Android SDK	69
Poznanie aplikacji przykładowych	74
Podsumowanie	75
Odwołania i inne źródła informacji	75
Rozdział 3 Pierwsza aplikacja na Androida	77
Testowanie środowiska programistycznego	77
Dodawanie aplikacji Snake do projektu	
w przestrzeni roboczej Eclipse	78
Tworzenie wirtualnego urządzenia (AVD)	
na potrzeby projektu Snake	80
Tworzenie konfiguracji uruchomieniowej projektu Snake	82
Uruchamianie aplikacji Snake w emulatorze	84
Budowanie pierwszej aplikacji na Androida	85
Tworzenie i konfiguracja nowego projektu aplikacji	86
Podstawowe pliki i katalogi aplikacji na Androida	88
Tworzenie AVD na potrzeby projektu	89
Tworzenie konfiguracji uruchomieniowych dla projektu	90
Uruchamianie aplikacji w emulatorze	91
Debugowanie aplikacji w emulatorze	93
Dodawanie mechanizmów rejestracji do aplikacji	97

	Dodawanie obsługi multimediów do aplikacji	99
	Dodawanie do aplikacji usług lokalizacyjnych	100
	Debugowanie aplikacji na fizycznym urządzeniu	103
	Podsumowanie	105
	Zasoby i inne źródła informacji	106
Część II	Kluczowe informacje	
	o konstrukcji aplikacji na Androida	107
Rozdział 4	Anatomia aplikacji na Androida	109
	Opanowanie najważniejszej terminologii	109
	Stosowanie kontekstu aplikacji	110
	Pobieranie kontekstu aplikacji	110
	Stosowanie kontekstu aplikacji	110
	Realizacja działań przy użyciu aktywności	111
	Cykl życia aktywności	112
	Zarządzanie zmianami aktywności przy użyciu intencji	117
	Praca z usługami	120
	Odbieranie oraz rozgłaszanie intencji	121
	Podsumowanie	122
	Odwołania i inne źródła informacji	123
Rozdział 5	Definiowanie aplikacji przy użyciu pliku manifestu	125
	Konfiguracja pliku manifestu	125
	Edycja pliku manifestu	126
	Zarządzanie tożsamością aplikacji	131
	Określanie wersji aplikacji	131
	Określanie nazwy oraz ikony aplikacji	131
	Określanie wymagań systemowych aplikacji	132
	Określanie konkretnych wersji SDK	132
	Ograniczenia związane z platformą	135
	Korzystanie z zewnętrznych bibliotek	137
	Rejestracja aktywności oraz innych komponentów aplikacji	138
	Określanie aktywności będącej głównym punktem	
	wejścia aplikacji przy użyciu filtra intencji	138
	Konfiguracja innych filtrów intencji	139
	Stosowanie uprawnień	140
	Rejestracja uprawnień wymaganych przez aplikację	140
	Rejestracja uprawnień przydzielanych innym aplikacjom	141
	Poznawanie innych ustawień podawanych w manifestcie	142
	Podsumowanie	142
	Odwołania i inne źródła informacji	143

Rozdział 6	Zarządzanie zasobami aplikacji	145
	Czym są zasoby?	145
	Przechowywanie zasobów aplikacji	146
	Zrozumienie hierarchii katalogów zasobów	146
	Typy wartości zasobów	147
	Przechowywanie zasobów różnych typów	149
	Programowy dostęp do zasobów	152
	Określanie wartości prostych zasobów w Eclipse	152
	Praca z zasobami	156
	Łańcuchy znaków	156
	Stosowanie łańcuchów z zasobów do formatowania tekstów	157
	Praca z tablicami łańcuchów znaków	159
	Praca z wartościami logicznymi	159
	Praca z liczbami całkowitymi	160
	Praca z kolorami	161
	Praca z wymiarami	162
	Praca z prostymi zasobami graficznymi	163
	Praca z obrazami	164
	Praca z animacjami	166
	Praca z menu	168
	Praca z plikami XML	170
	Praca z nieprzetworzonymi plikami	171
	Odwołania do zasobów	171
	Praca z układami	173
	Praca ze stylami	178
	Praca z tematami	181
	Odwołania do zasobów systemowych	181
	Podsumowanie	182
	Odwołania i inne źródła informacji	183
Część III	Kluczowe zagadnienia	
	projektowania interfejsów użytkownika	185
Rozdział 7	Prezentacja wizualnych elementów interfejsu użytkownika	187
	Prezentacja widoków i układów	187
	Prezentacja widoków	187
	Prezentacja kontroltek	188
	Prezentacja układów	188
	Wyświetlanie tekstów przy użyciu TextView	189
	Konfiguracja układu oraz określanie wymiarów	190
	Umieszczanie w tekście kontekstowych odnośników	191
	Pobieranie danych od użytkowników	192
	Pobieranie danych przy użyciu kontroltek EditText	193
	Kontrolka Spinner — zapewnianie możliwości wyboru	199

Stosowanie przycisków, pól wyboru oraz pól opcji	200
Stosowanie zwyczajnych przycisków	201
Stosowanie pól wyboru i przełączników	203
Stosowanie kontrolki RadioGroup oraz RadioButton	204
Pobieranie daty i godziny	206
Prezentacja danych przy użyciu wskaźników	208
Prezentacja postępów przy użyciu paska postępu	208
Modyfikacja postępu przy użyciu kontrolki SeekBar	211
Wyświetlanie oceny przy użyciu kontrolki RatingBar	212
Prezentacja upływu czasu przy użyciu minutnika	213
Wyświetlanie czasu	213
Prezentowanie menu opcji oraz menu kontekstowego	214
Edycja menu opcji	215
Obsługa zdarzeń generowanych przez użytkownika	219
Odbieranie informacji o zmianach trybu dotyku	219
Odbieranie zdarzeń dotyczących całego ekranu	221
Obsługa zdarzeń naciśnięcia i przytrzymania	222
Reagowanie na zmiany wybranej kontrolki	223
Stosowanie okien dialogowych	224
Różne typy okien dialogowych	224
Śledzenie cyklu życia okna dialogowego	225
Tworzenie własnych okien dialogowych	227
Stosowanie stylów	227
Stosowanie tematów	229
Podsumowanie	231
Rozdział 8 Projektowanie interfejsów użytkownika z użyciem układów ... 233	
Tworzenie interfejsów użytkownika w systemie Android	233
Definiowanie układów w zasobach aplikacji	233
Programowe tworzenie układów	235
Organizacja interfejsu użytkownika aplikacji	238
Zrozumienie różnic między View i ViewGroup	238
Stosowanie wbudowanych układów	243
Układ FrameLayout	244
Układ LinearLayout	247
Układ RelativeLayout	247
Układ TableLayout	251
Stosowanie wielu układów jednocześnie	254
Stosowanie wbudowanych klas pojemników	255
Pojemniki działające w oparciu o dane	255
Organizacja interfejsu użytkownika przy wykorzystaniu zakładek ...	260
Dodawanie możliwości przewijania	264
Prezentacja innych rodzajów pojemników	265
Podsumowanie	266

Rozdział 9 Rysowanie i korzystanie z animacji	267
Rysowanie na ekranie	267
Stosowanie obiektów Canvas oraz Paint	267
Wyświetlanie tekstów	272
Stosowanie domyślnych czcionek i ich krojów	272
Stosowanie własnych czcionek	273
Określanie wymiarów tekstu na ekranie	275
Wyświetlanie i operacje na grafice rastrowej	275
Wyświetlanie grafiki rastrowej	275
Skalowanie grafiki rastrowej	275
Macierzowe przekształcenia grafiki rastrowej	275
Wyświetlanie i operacje na kształtach	277
Definiowanie kształtów w postaci zasobów XML	277
Programowe definiowanie kształtów	278
Rysowanie różnych kształtów	279
Stosowanie animacji	285
Stosowanie animacji poklatkowych	286
Stosowanie animacji przejść	288
Podsumowanie	294

Część IV Najczęściej stosowane interfejsy programowania aplikacji na Androida295

Rozdział 10 Obsługa danych i składowanie	297
Korzystanie z preferencji aplikacji	297
Tworzenie właściwości prywatnych i wspólnych	298
Przeszukiwanie i odczyt preferencji	299
Dodawanie, aktualizacja oraz usuwanie preferencji	299
Odnajdywanie danych preferencji w systemie plików Androida	300
Obsługa plików i katalogów	301
Poznawanie zawartości katalogów aplikacji	301
Korzystanie z innych katalogów i plików w systemie Android	304
Przechowywanie danych strukturalnych w bazach danych SQLite	305
Tworzenie baz danych SQLite	307
Tworzenie, aktualizacja oraz usuwanie rekordów z baz danych ...	309
Wykonywanie zapytań w bazach danych SQLite	312
Zamykanie i usuwanie baz danych SQLite	317
Projektowanie trwałych baz danych	318
Kojarzenie danych z interfejsem użytkownika aplikacji	320
Podsumowanie	325
Odwolania oraz dodatkowe źródła informacji	326

Rozdział 11 Udostępnianie danych pomiędzy aplikacjami przy użyciu dostawców treści	327
Prezentacja dostawców treści	327
Stosowanie dostawcy treści MediaStore	328
Stosowanie dostawcy danych CallLog	329
Stosowanie dostawcy treści Browser	331
Stosowanie dostawcy treści Contacts	332
Stosowanie dostawcy treści UserDictionary	335
Stosowanie dostawcy treści Settings	336
Modyfikacja danych dostawców treści	336
Dodawanie rekordów	336
Aktualizacja rekordów	337
Usuwanie rekordów	338
Wzbogacanie możliwości aplikacji przy użyciu dostawców treści	338
Dostęp do zgromadzonych obrazów	339
Działanie jako dostawca treści	344
Implementacja interfejsu dostawcy treści	344
Definiowanie identyfikatora URI danych	345
Definiowanie kolumn danych	345
Implementacja metod dostawcy treści	346
Modyfikacja pliku manifestu	351
Aktywne foldery	352
Podsumowanie	355
Odwołania i inne źródła informacji	355
Rozdział 12 Obsługa operacji sieciowych	357
Podstawy operacji sieciowych na urządzeniach przenośnych	357
Dostęp do internetu (HTTP)	358
Odczytywanie danych z sieci	358
Stosowanie klasy HttpURLConnection	359
Przetwarzanie kodu XML pobieranego z sieci	360
Przetwarzanie asynchroniczne	362
Stosowanie klasy AsyncTask	362
Zastosowanie wątków do obsługi operacji sieciowych	364
Wyświetlanie obrazów pobieranych z internetu	365
Pobieranie statusu połączenia sieciowego	367
Podsumowanie	369
Odwołania i inne źródła informacji	369
Rozdział 13 Obsługa WWW	371
Przeglądanie WWW przy użyciu WebView	371
Projektowanie układu z kontrolką WebView	372
Wyświetlanie treści w kontrolce WebView	372
Wzbogacanie możliwości kontrolki WebView	374

Tworzenie rozszerzeń przy użyciu WebKit	378
Przegląd interfejsów mechanizmu WebKit	378
Tworzenie aplikacji internetowych w systemie Android	378
Stosowanie technologii Flash	383
Korzystanie z aplikacji napisanych w technologii Flash	384
Tworzenie aplikacji na Androida w technologii AIR	384
Podsumowanie	385
Odwołania i dodatkowe źródła informacji	386
Rozdział 14 Korzystanie z usług lokalizacyjnych	387
Stosowanie usług GPS	387
Korzystanie z możliwości GPS w aplikacji	388
Określanie położenia	388
Określanie położenia w emulatorze	390
Geokodowanie lokalizacji	391
Prezentacja lokalizacji na mapie	394
Intencje wykorzystujące mapy	394
Kontrolka mapy	396
Pobieranie testowego klucza API	397
Przesuwanie mapy	399
Powiększanie mapy	400
Zaznaczanie miejsc	401
Inne możliwości, jakie zapewniają usługi lokalizacyjne	406
Podsumowanie	407
Odwołania i inne źródła informacji	407
Rozdział 15 Obsługa multimediów	409
Korzystanie z multimediów	409
Korzystanie ze zdjęć	410
Rejestracja zdjęć przy użyciu aparatu fotograficznego	410
Konfiguracja trybu działania aparatu	415
Współdzielenie zdjęć	416
Stosowanie obrazów jako tapet	417
Korzystanie z wideo	418
Rejestracja wideo	418
Odtwarzanie obrazu wideo	419
Rejestracja i odtwarzanie dźwięków	421
Rejestracja dźwięków	421
Odtwarzanie plików dźwiękowych	423
Współdzielenie plików dźwiękowych	424
Poszukiwanie multimediów	425
Korzystanie z dzwonek	425
Podsumowanie	426
Odwołania i inne źródła informacji	426

Rozdział 16 Obsługa telefonii	427
Korzystanie z narzędzi telefonii	427
Uzyskiwanie uprawnień do pobierania statusu telefonu	428
Pobieranie informacji o stanie rozmowy	428
Pobieranie informacji o usłudze	430
Monitorowanie siły sygnału i szybkości połączenia	431
Obsługa numerów telefonicznych	431
Korzystanie z krótkich wiadomości tekstowych	432
Uzyskiwanie uprawnień do wysyłania SMS-ów	433
Wysyłanie wiadomości tekstowych	433
Odbieranie SMS-ów	435
Nawiązywanie i odbieranie połączeń telefonicznych	437
Nawiązywanie połączeń telefonicznych	437
Odbieranie połączeń telefonicznych	438
Podsumowanie	439
Odwołania i inne źródła informacji	440
Rozdział 17 Obsługa grafiki 3D przy użyciu OpenGL ES	441
Stosowanie OpenGL ES	441
Stosowanie OpenGL ES w Androidzie	442
Zapewnianie zgodności urządzeń	443
Korzystanie z biblioteki OpenGL ES w Androidzie	443
Ręczna obsługa OpenGL ES	444
Tworzenie obiektu SurfaceView	445
Uruchamianie wątku dla operacji OpenGL ES	446
Inicjalizacja biblioteki EGL	448
Inicjalizacja biblioteki GL	450
Rysowanie na ekranie	450
Rysowanie obiektów trójwymiarowych	452
Rysowanie wierzchołków	452
Kolorowanie wierzchołków	453
Rysowanie bardziej złożonych obiektów	453
Oświetlanie sceny	455
Dodawanie tekstur do obiektów	457
Interakcja z kontrolkami i zdarzeniami systemu Android	459
Zapewnianie komunikacji pomiędzy wątkiem OpenGL i wątkiem aplikacji	460
Zapewnianie możliwości przesyłania danych z wątku aplikacji do wątku OpenGL	461
Zwalnianie zasobów OpenGL ES	464
Stosowanie klasy GLSurfaceView (prosty sposób na OpenGL ES)	464
Stosowanie OpenGL ES 2.0	467
Konfiguracja aplikacji w celu korzystania z OpenGL 2.0	468
Pobieranie powierzchni OpenGL ES 2.0	468
Podsumowanie	472
Odwołania i inne źródła informacji	473

Rozdział 18 Stosowanie Android NDK	475
Określanie, kiedy warto użyć NDK	475
Instalacja Android NDK	476
Prezentacja Android NDK	477
Uruchamianie aplikacji przykładowej	477
Tworzenie własnego projektu NDK	478
Wywoływanie kodu rodzimego z poziomu Javy	478
Obsługa parametrów i wartości wynikowych	480
Stosowanie wyjątków w kodzie rodzimym	480
Poprawa wydajności generowania grafiki	482
Podsumowanie	483
Odwołania i inne źródła informacji	484
Rozdział 19 Korzystanie z opcjonalnych komponentów sprzętowych	485
Interakcja z komponentami sprzętowymi urządzenia	485
Korzystanie z czujników	486
Korzystanie z różnych czujników	487
Uzyskiwanie dostępu do czujnika	488
Odczyt danych z czujnika	488
Kalibracja czujników	490
Określanie położenia	490
Odnajdywanie północy	491
Korzystanie z Wi-Fi	491
Korzystanie z technologii Bluetooth	494
Sprawdzanie dostępności komponentów sprzętowych	495
Włączanie komunikacji Bluetooth	495
Poszukiwanie sparowanych urządzeń	495
Odkrywanie urządzeń	496
Nawiązywanie połączenia pomiędzy dwoma urządzeniami	496
Monitorowanie stanu baterii	497
Podsumowanie	500
Odwołania i inne źródła informacji	501
Część V Dodatkowe zasady projektowania aplikacji na Androida	503
Rozdział 20 Stosowanie powiadomień	505
Powiadamianie użytkownika	505
Powiadomienia na pasku stanu	506
Stosowanie usługi NotificationManager	507
Tworzenie prostego powiadomienia z ikoną	507
Obsługa kolejki powiadomień	508
Aktualizacja powiadomień	510
Usuwanie powiadomień	511

Stosowanie wibracji	511
Błyskanie	512
Stosowanie dźwięków	514
Określanie postaci powiadomień	515
Projektowanie użytecznych powiadomień	517
Podsumowanie	517
Odwołania i inne źródła informacji	518
Rozdział 21 Stosowanie usług	519
Określanie, kiedy należy użyć usługi	519
Prezentacja cyklu życia usługi	520
Tworzenie usługi	521
Kontrola działania usługi	526
Implementacja zdalnego interfejsu	527
Implementacja klasy Parcelable	529
Podsumowanie	532
Odwołania i inne źródła informacji	532
Rozdział 22 Poszerzanie zasięgu aplikacji	533
Wzbogacanie aplikacji	533
Tworzenie widżetów aplikacji	534
Tworzenie widżetu aplikacji	535
Instalacja widżetu	543
Hosty widżetów	544
Animowane tapety	545
Tworzenie animowanej tapety	545
Instalacja animowanej tapety	549
Obsługa konkretnego typu danych	549
Określanie akcji intencji oraz typów MIME	551
Implementacja aktywności do obsługi intencji	551
Rejestracja filtra intencji	552
Zapewnianie możliwości przeszukiwania treści aplikacji	553
Zapewnianie możliwości przeszukiwania wewnątrz aplikacji	555
Zapewnianie możliwości korzystania z wyszukiwania globalnego	562
Korzystanie z aktywnych folderów	565
Tworzenie aktywnego folderu	566
Instalacja aktywnego folderu	570
Podsumowanie	571
Odwołania i inne źródła informacji	572
Rozdział 23 Zarządzanie kontami użytkownika oraz synchronizacja jego danych	573
Zarządzanie kontami przy użyciu menedżera kont	573
Synchronizacja danych przy użyciu adapterów synchronizacji	575
Korzystanie z usług tworzenia kopii zapasowych	576

Wybór zdalnej usługi tworzenia kopii	576
Implementacja agenta archiwizacji	577
Tworzenie kopii i odtwarzanie danych aplikacji	581
Podsumowanie	582
Odwołania i inne źródła informacji	582
Rozdział 24 Zaawansowane sposoby obsługi wprowadzania danych	585
Stosowanie różnych metod wprowadzania tekstów	585
Stosowanie klawiatur programowych	586
Korzystanie z przewidywania tekstu i słowników użytkownika	589
Przedstawienie szkieletu ułatwień dostępu	589
Korzystanie z usług rozpoznawania mowy	591
Korzystanie z usług odczytywania tekstów	593
Korzystanie z gestów	596
Wykrywanie ruchów w widokach	596
Obsługa popularnych gestów wykonywanych jednym palcem	597
Obsługa gestów wielodotyku	604
Zapewnianie bardziej naturalnego wyglądu gestów	607
Korzystanie z manipulatora kulowego	607
Obsługa zmian orientacji ekranu	607
Podsumowanie	610
Odwołania i inne źródła informacji	611
Rozdział 25 Przygotowywanie aplikacji dla różnych konfiguracji	
 sprzętowych i języków	613
Maksymalizowanie zgodności aplikacji	613
Projektowanie interfejsu użytkownika	
pod kątem zachowania zgodności	615
Obsługa konkretnych typów ekranów	617
Korzystanie z grafiki typu Nine-Patch Stretchable Graphics	617
Stosowanie zasady kwadratu roboczego	621
Stosowanie zasobów alternatywnych	622
Kwalifikatory zasobów alternatywnych	622
Tworzenie zasobów na potrzeby działania	
przy różnej orientacji ekranu	628
Programowe stosowanie zasobów alternatywnych	629
Określanie efektywnej organizacji zasobów	630
Umieędzynarodawanie aplikacji	632
Umieędzynarodawanie przy wykorzystaniu zasobów alternatywnych ..	632
Programowa obsługa ustawień lokalnych	637
Obsługa różnych konfiguracji urządzeń	638
Obsługa konfiguracji sprzętowych	639
Określanie wersji Android SDK, na jakich działa aplikacja	640
Podsumowanie	643
Odwołania i inne źródła informacji	643

Część VI Wdrażanie aplikacji	645
Rozdział 26 Proces tworzenia oprogramowania mobilnego	647
Prezentacja procesu tworzenia oprogramowania mobilnego	647
Wybór metodologii tworzenia oprogramowania	648
Zrozumienie niebezpieczeństw metody kaskadowej	648
Zrozumienie znaczenia powtarzania	649
Gromadzenie wymagań aplikacji	649
Określanie wymagań projektowych	650
Tworzenie przypadków użycia oprogramowania mobilnego	652
Dołączanie wymagań innych podmiotów	652
Zarządzanie bazą danych urządzeń	653
Szacowanie ryzyka związanego z projektem	656
Określanie urządzeń docelowych	656
Pozyskiwanie urządzeń docelowych	658
Określanie możliwości zaspokojenia wymagań aplikacji	659
Rozumienie ryzyka związanego z zapewnieniem jakości	659
Pisanie ważnej dokumentacji projektowej	661
Tworzenie planów testowania na potrzeby kontroli jakości	662
Dostarczanie dokumentacji wymaganej przez inne podmioty	662
Dokumentacja na potrzeby utrzymania i przenoszenia	662
Korzystanie z systemów zarządzania konfiguracją	663
Wybór systemu zarządzania kodem źródłowym	663
Implementacja działającego systemu numeracji wersji aplikacji	663
Projektowanie aplikacji mobilnych	664
Znajomość ograniczeń urządzeń przenośnych	664
Poznanie wspólnych architektur aplikacji mobilnych	664
Projektowanie aplikacji pod kątem jej rozszerzania i pielęgnacji	665
Projektowanie pod kątem możliwości współdziałania aplikacji	667
Tworzenie aplikacji na urządzenia przenośne	667
Testowanie aplikacji na urządzenia przenośne	668
Wdrażanie aplikacji mobilnych	668
Określanie rynków docelowych	669
Wsparcie i pielęgnacja oprogramowania mobilnego	669
Śledzenie i weryfikacja informacji o awariach	669
Testowanie aktualizacji oprogramowania układowego	670
Prowadzenie odpowiedniej dokumentacji aplikacji	670
Wprowadzanie zmian na działającym serwerze	670
Określanie możliwości przenoszenia aplikacji obciążonego niewielkim ryzykiem	670
Podsumowanie	671
Odwołania i dodatkowe źródła informacji	671

Rozdział 27 Projektowanie i tworzenie niezawodnych aplikacji na Androida	673
Najlepsze praktyki projektowania niezawodnych aplikacji mobilnych ...	673
Zaspokajanie wymagań użytkowników urządzeń przenośnych	674
Projektowanie interfejsu użytkownika aplikacji mobilnych	675
Projektowanie stabilnych i szybko reagujących aplikacji mobilnych	676
Projektowanie bezpiecznych aplikacji mobilnych	678
Projektowanie aplikacji w celu maksymalizacji zysków	679
Korzystanie ze standardów projektowania aplikacji opracowanych przez innych	680
Projektowanie aplikacji pod kątem prostoty ich utrzymania i aktualizacji	680
Projektowanie aplikacji przy wykorzystaniu narzędzi Androida	682
Unikanie głupich błędów podczas projektowania aplikacji na Androida	683
Najlepsze praktyki stosowane przy tworzeniu niezawodnego oprogramowania mobilnego	683
Określanie procesu produkcyjnego dostosowanego do tworzenia oprogramowania mobilnego	684
Wczesne i częste testowanie możliwości wykonania projektu	684
Stosowanie standardów kodowania, weryfikacji i testów jednostkowych w celu poprawienia jakości kodu	685
Obsługa usterek występujących na jednym urządzeniu	688
Korzystanie z narzędzi Androida przy pisaniu aplikacji	689
Unikanie głupich błędów podczas tworzenia aplikacji na Androida ...	689
Podsumowanie	689
Odwołania i inne źródła informacji	690
Rozdział 28 Testowanie aplikacji na Androida	691
Najlepsze praktyki testowania oprogramowania na urządzenia przenośne	691
Projektowanie systemu rejestracji defektów na potrzeby tworzenia oprogramowania mobilnego	691
Zarządzanie środowiskiem testowym	693
Maksymalizacja pokrycia testów	696
Stosowanie narzędzi do testowania aplikacji na Androida	704
Unikanie głupich błędów podczas testowania aplikacji na Androida	705
Korzystanie z usług testowania aplikacji	706
Podsumowanie	706
Odwołania i inne źródła informacji	706
Rozdział 29 Sprzedawanie aplikacji na Androida	709
Wybór odpowiedniego modelu dystrybucji	709
Przygotowywanie aplikacji do publikacji	710
Przygotowanie kodu do utworzenia pakietu instalacyjnego	711

Tworzenie pakietu aplikacji i jego podpisywanie	713
Testowanie publikowanej wersji pakietu aplikacji	716
Certyfikacja aplikacji na Androida	716
Dystrybucja aplikacji	717
Sprzedawanie aplikacji w Android Market	717
Sprzedawanie aplikacji na własnym serwerze	724
Korzystanie z innych sposobów sprzedawania aplikacji	725
Ochrona własności intelektualnej	726
Pobieranie opłat od użytkowników	727
Podsumowanie	727
Odwołania i inne źródła informacji	728

Dodatki 729

Dodatek A Krótki przewodnik po emulatorze Androida 731

Symulacja rzeczywistości — przeznaczenie emulatora	731
Korzystanie z różnych urządzeń wirtualnych (AVD)	733
Stosowanie narzędzia Android SDK and AVD Manager	734
Tworzenie AVD	735
Uruchamianie emulatora z użyciem konkretnego AVD	740
Konfiguracja opcji uruchomieniowych emulatora	741
Uruchamianie emulatora w celu wykonania aplikacji	741
Uruchamianie emulatora z poziomu programu Android SDK and AVD Manager	743
Konfiguracja położenia GPS w emulatorze	744
Nawiązywanie połączeń telefonicznych pomiędzy dwoma egzemplarzami emulatora	745
Przesyłanie SMS-ów pomiędzy dwoma egzemplarzami emulatora	747
Interakcja z emulatorem z poziomu konsoli	749
Wykorzystanie konsoli do symulowania odbieranych połączeń	750
Stosowanie konsoli do symulowania wiadomości SMS	751
Stosowanie konsoli do przesyłania współrzędnych GPS	752
Stosowanie konsoli do monitorowania transmisji sieciowych	752
Stosowanie konsoli do modyfikowania ustawień zasilania	752
Inne polecenia konsoli emulatora	753
Korzystanie z emulatora dla zabawy	754
Ograniczenia emulatora	755

Dodatek B Krótki przewodnik po DDMS 757

Korzystanie z DDMS jako niezależnej aplikacji oraz w Eclipse	757
Szybka prezentacja kluczowych możliwości DDMS	758
Obsługa procesów	760
Dołączanie debugera do aplikacji	760
Monitorowanie aktywności wątku aplikacji	761
Żądanie oczyszczenia pamięci	762

Monitorowanie operacji wykonywanych na sterście	762
Monitorowanie przydzielanej pamięci	763
Zatrzymywanie procesu	763
Zarządzanie plikami	764
Przeglądanie systemu plików w emulatorze lub na urządzeniu	764
Kopiowanie plików z emulatora lub urządzenia	765
Kopiowanie plików do emulatora lub urządzenia	765
Usuwanie plików na emulatorze lub urządzeniu	765
Stosowanie zakładki Emulator Control	766
Symulowanie przychodzących połączeń telefonicznych	766
Symulowanie nadsyłanych wiadomości SMS	766
Przesyłanie współrzędnych geograficznych	767
Korzystanie z mechanizmów rejestracji komunikatów	767
Wykonywanie zrzutów ekranu z emulatora lub urządzenia	768

Dodatek C Krótki przewodnik po ADB 771

Wyświetlanie listy podłączonych urządzeń i uruchomionych emulatorów	771
Kierowanie poleceń ADB do konkretnych urządzeń	772
Uruchamianie i zatrzymywanie serwera ADB	772
Zatrzymywanie procesu serwera ADB	773
Uruchamianie i sprawdzanie procesu serwera ADB	773
Wykonywanie poleceń powłoki	773
Wydawanie pojedynczych poleceń powłoki	773
Stosowanie sesji	773
Zastosowanie powłoki do uruchamiania i zatrzymywania emulatora	774
Kopiowanie plików	774
Wysyłanie plików do urządzenia lub emulatora	775
Pobieranie plików z urządzenia lub emulatora	775
Instalowanie i usuwanie aplikacji	775
Instalowanie aplikacji	776
Ponowna instalacja aplikacji	776
Deinstalacja aplikacji	776
Korzystanie z narzędzia LogCat	776
Wyświetlanie wszystkich zarejestrowanych komunikatów	777
Prezentowanie daty i godziny	777
Filtrowanie prezentowanych informacji	777
Czyszczenie listy komunikatów	779
Przekierowywanie prezentowanych informacji do pliku	779
Dostęp do innych dzienników	779
Kontrola usługi kopii zapasowej	779
Wymuszanie sporządzenia kopii zapasowej	780
Wymuszanie odtworzenia danych	780
Usuwanie kopii zapasowej	781

Generacja raportu o błędach	781
Korzystanie z powłoki do wykonywania operacji na bazach danych SQLite	781
Korzystanie z powłoki do testowania aplikacji w warunkach zwiększonego obciążenia	781
Małpia zabawa z aplikacją	782
Odbieranie informacji o działaniach programu monkey	782
Określanie czynności symulowanych przez program monkey	782
Powtarzanie tych samych operacji	784
Kontrola działania programu	784
Dokładniejsze poznanie programu monkey	785
Instalowanie niestandardowych programów binarnych	785
Poznanie innych poleceń ADB	786
Dodatek D Sztuczki i triki podczas korzystania z Eclipse IDE	787
Organizacja przestrzeni roboczej	787
Integracja z usługami kontroli wersji	787
Zmiana położenia zakładek w perspektywie	788
Maksymalizacja okien	788
Minimalizacja okien	788
Umieszczanie okien obok siebie	789
Wyświetlanie dwóch fragmentów tego samego pliku	789
Zamykanie niepotrzebnych zakładek	789
Zachowanie kontroli nad oknami	789
Tworzenie własnych filtrów dziennika	790
Pisanie kodu w języku Java	790
Automatyczne uzupełnianie	790
Formatowanie kodu	791
Tworzenie nowych klas	791
Tworzenie nowych metod	791
Organizacja instrukcji import	792
Modyfikacja niemal wszystkich nazw	792
Refaktoryzacja kodu	793
Stosowanie narzędzia Extract Local Variable	793
Reorganizacja kodu	794
Tworzenie dokumentacji Javadoc	795
Wyjaśnianie tajemniczych błędów	795
Dodatek E Przewodnik po SQLite dla początkujących	797
Podstawowe czynności obsługi baz SQLite	797
Stosowanie programu sqlite3	798
Uruchamianie powłoki ADB	798
Nawiązywanie połączenia z bazą danych SQLite	799
Przeglądanie baz danych	799
Importowanie oraz eksportowanie baz danych i ich zawartości	800

Wykonywanie poleceń SQL w wierszu poleceń	802
Stosowanie innych poleceń sqLite3	803
Znajomość ograniczeń baz danych SQLite	803
Nauka na przykładach — baza ocen uczniów	804
Projektowanie struktury bazy danych ocen	805
Tworzenie prostej tabeli z kolumną AUTOINCREMENT	805
Zapisywanie danych w tabelach	806
Pobieranie informacji z tabel przy użyciu polecenia SELECT	806
Stosowanie kluczy obcych oraz złożonych kluczy głównych	807
Modyfikowanie i aktualizacja danych	808
Pobieranie danych z kilku tabel przy użyciu złączeń	809
Stosowanie kolumn o obliczanych wartościach	810
Wyliczanie wartości kolumn przy użyciu podzapytań	811
Usuwanie tabel	811
Skorowidz	813

Pierwsza aplikacja na Androida

Aktualnie Czytelnik powinien już posiadać na własnym komputerze zainstalowane i działające środowisko do pisania aplikacji na platformę Androida. Można mieć także nadzieję, że oprócz tego Czytelnik posiada także jakieś urządzenie z Androidem. Nadszedł zatem czas, by napisać swój pierwszy kod na tę platformę.

W tym rozdziale Czytelnik dowie się, jak należy dodawać i tworzyć nowe projekty aplikacji na Androida w środowisku Eclipse oraz jak sprawdzić, czy używane środowisko programistyczne zostało prawidłowo skonfigurowane. Oprócz tego w tym rozdziale Czytelnik napisze i przetestuje swoją pierwszą aplikację na Androida, uruchamiając ją zarówno w emulatorze, jak i na faktycznym urządzeniu.

Testowanie środowiska programistycznego

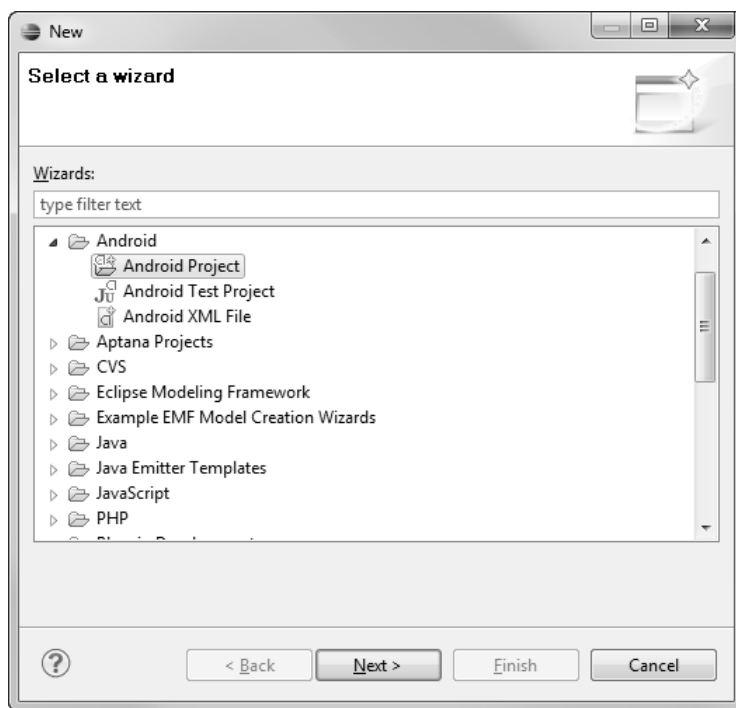
Najlepszym sposobem, by upewnić się, że używane środowisko programistyczne zostało skonfigurowane prawidłowo, jest skorzystanie z jakiejś gotowej aplikacji i przeprowadzenie próby jej uruchomienia. Z łatwością można do tego celu wykorzystać jedną z wielu przykładowych aplikacji dostarczanych wraz z Android SDK i umieszczonych w katalogu */samples*.

Wśród wielu przykładów aplikacji dostarczanych wraz z Android SDK można znaleźć klasyczną grę o nazwie Snake (wąż). Aby ją zbudować i uruchomić, w pierwszej kolejności należy utworzyć w przestrzeni roboczej środowiska Eclipse nowy projekt aplikacji na Androida, stworzyć odpowiedni profil wirtualnego urządzenia (AVD) i przygotować konfigurację uruchomieniową dla projektu. Po prawidłowym skonfigurowaniu wszystkich elementów środowiska będzie można zbudować aplikację i uruchomić ją na emulatorze bądź rzeczywistym urządzeniu.

Dodawanie aplikacji Snake do projektu w przestrzeni roboczej Eclipse

Pierwszą rzeczą, jaką należy zrobić, jest dodanie projektu *Snake* do przestrzeni roboczej Eclipse. W tym celu należy wykonać następujące czynności:

1. Wybrać opcję *File/New/Project*.
2. Rozwinąć grupę *Android*, a następnie wybrać kreatora *Android Project* (patrz rysunek 3.1).



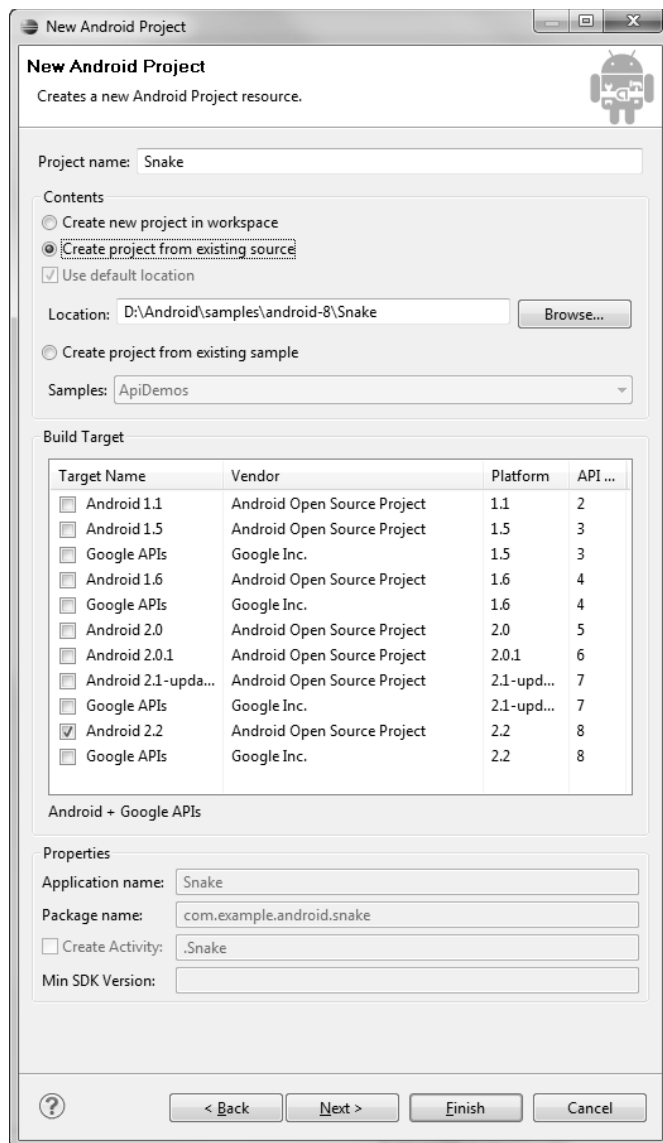
Rysunek 3.1. Tworzenie nowego projektu aplikacji na Androida

Wskazówka

Kiedy już raz wybierzemy kreatora *Android Project*, kolejne projekty będzie można tworzyć, wybierając z menu opcję *File/New/Android Project*.

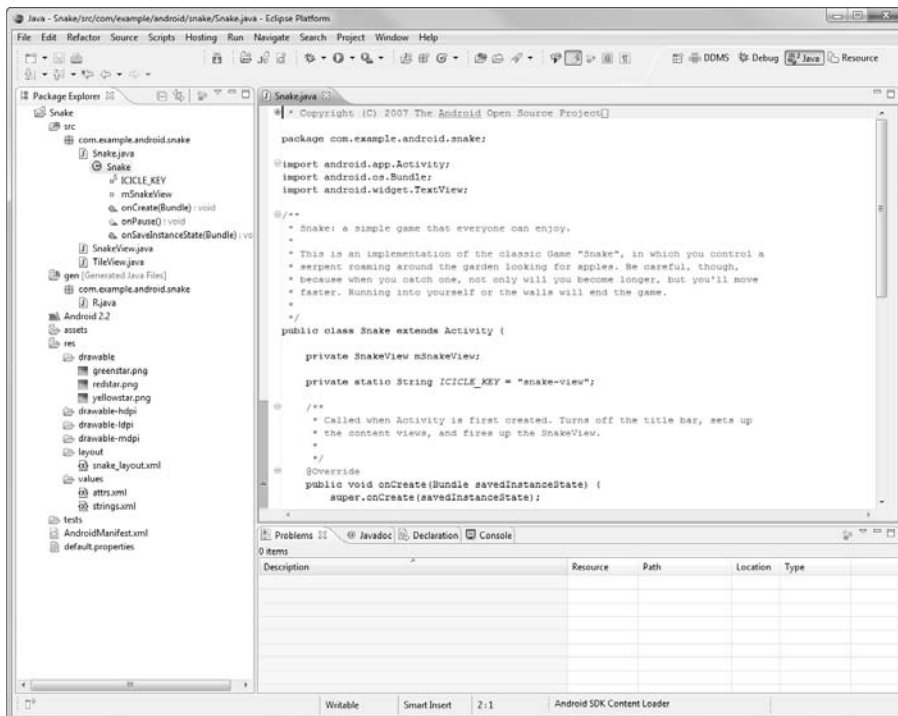
3. W sekcji *Content* zmienić wybraną opcję *Create new project in workspace* na *Create project from existing sources*.
4. Kliknąć przycisk *Browse* i przeglądnąć zawartość katalogu z programami przykładowymi dostarczanymi wraz z Android SDK.

- Wybrać katalog *Snake*. Wszystkie pola kreatora powinny zostać wypełnione informacjami odczytanymi z pliku manifestu (patrz rysunek 3.2). Dodatkowo warto sprawdzić ustawienie *Build Target* i wybrać w nim tę wersję platformy Android, w jakiej mamy zamiar uruchamiać aplikację.



Rysunek 3.2. Szczegółowe informacje na temat projektu Snake

6. Kliknąć przycisk *Finish*. Kiedy to zrobimy, projekt *Snake* powinien zostać otwarty w środowisku Eclipse (patrz rysunek 3.3).



Rysunek 3.3. Pliki projektu Snake

ostrzeżenie

Od czasu do czasu może się zdarzyć, że Eclipse wyświetli następujący komunikat o błędzie: *Project „Snake” is missing required source folder: „gen”*¹. Jeśli tak się zdarzy, wystarczy odnaleźć w projekcie i usunąć plik *R.java* umieszczony w katalogu */gen*. Plik *R.java* jest generowany automatycznie, a po jego usunięciu problem powinien zniknąć.


Tworzenie wirtualnego urządzenia (AVD) na potrzeby projektu Snake

Kolejną czynnością, jaką należy wykonać, jest utworzenie wirtualnego urządzenia, które będzie opisywać typ urządzenia, jakie chcemy symulować w emulatorze podczas uruchamiania aplikacji; można przy tym określić, w jaką wersję platformy Android ma

¹ W projekcie „Snake” brakuje wymaganego katalogu źródłowego: „gen” — *przyp. tłum.*

zostać wyposażone urządzenie. Można określać także różne wielkości ekranów oraz ich orientacje, zażądać, by symulowane urządzenie było wyposażone w kartę SD, oraz podać jej pojemność.

Na potrzeby tego przykładu wystarczy AVD z domyślną wersją platformy Android 2.2. Poniżej opisane zostały czynności, jakie należy wykonać w celu utworzenia prostego urządzenia wirtualnego (AVD):

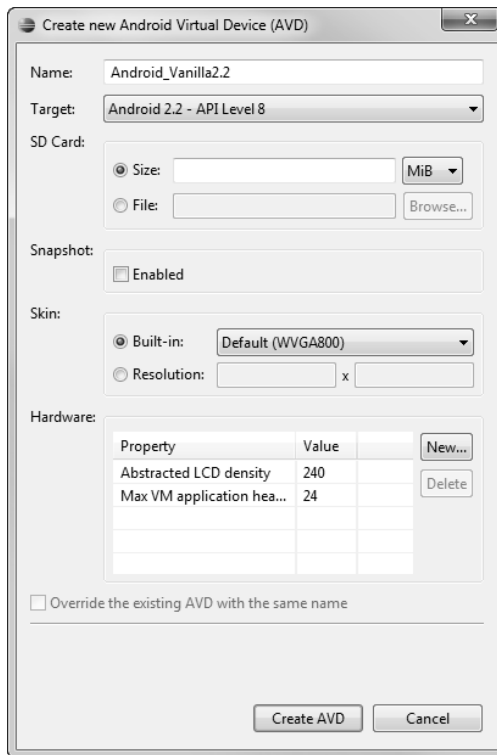
1. Z poziomu Eclipse uruchomić program Android Virtual Device, klikając umieszczony na pasku narzędzi przycisk z niewielką ikoną Androida i strzałką skierowaną w dół — . Jeśli nie można znaleźć tej ikony, to program menedżera można także uruchomić za pośrednictwem opcji dostępnych w menu *Window* środowiska Eclipse.
2. W menu *Virtual Devices* kliknąć przycisk *New*.
3. Podać nazwę tworzonego AVD. Ponieważ mamy zamiar wykorzystać wszystkie domyślne ustawienia, nadamy mu nazwę *Android_Vanilla2.2*.
4. Wybrać urządzenie docelowe. Ponieważ chcemy stworzyć zwyczajne urządzenie z Androidem 2.2, z rozwijalnej listy należy wybrać opcję *Android 2.2*.
5. Określić pojemność karty SD. Można ją podać w kilobajtach lub megabajtach. Obszar ten jest przydzielany z dostępnej przestrzeni na dysku twardym komputera. Warto wybrać jakąś rozsądną wartość, na przykład 1 GB (1024 MB). Jeśli Czytelnikowi brakuje wolnego miejsca na dysku bądź jeśli wiadomo, że nie trzeba będzie testować przechowywania danych aplikacji na karcie SD, to można utworzyć kartę o znacznie mniejszej pojemności, na przykład 64 MB.
6. Wybrać wygląd symulowanego urządzenia, tak zwaną „skórkę” (ang. *skin*). Ta opcja pozwala określać różne rozdzielczości ekranu symulowanego urządzenia. W naszym przypadku zastosujemy skórkę *WVGA800*. Chyba najdokładniej odpowiada ona najbardziej popularnym telefonom z Androidem, takim jak HTC Negus One lub Evo 4G, które właśnie leżą przede mną na biurku.

Ustawienia projektu zostały przedstawione na rysunku 3.4.

Wskazówka

Choć w tej książce skonfigurowaliśmy AVD tak, by emulowane urządzenie miało wysoką rozdzielczość, to jednak Czytelnik może wybrać inne ustawienia, odpowiadające urządzeniu, na jakim zamierza on uruchamiać aplikację.

7. Kliknąć przycisk *Create AVD* i poczekać na wykonanie operacji.
8. Kliknąć przycisk *Finish*. Ponieważ menedżer AVD formatuje pamięć przydzieloną na potrzeby karty SD symulowanego urządzenia, tworzenie wirtualnych urządzeń z kartami pamięci może chwilę potrwać.



Rysunek 3.4. Tworzenie nowego AVD w środowisku Eclipse

Wskazówka

Wirtualne urządzenia AVD można także tworzyć przy użyciu narzędzi wchodzących w skład Android SDK wybieranych z poziomu wiersza poleceń.

Więcej informacji na temat tworzenia różnych typów urządzeń AVD można znaleźć w dodatku A, „Krótki przewodnik po emulatorze Androida”.

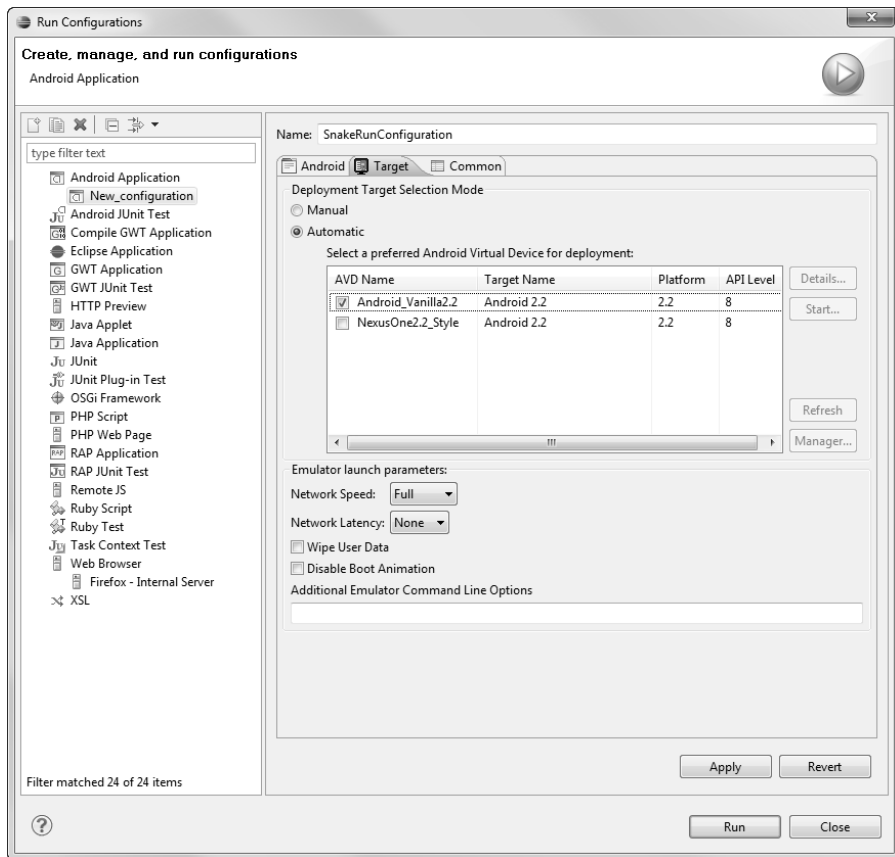
Tworzenie konfiguracji uruchomieniowej projektu Snake

Kolejną czynnością jest utworzenie konfiguracji uruchomieniowej w środowisku Eclipse. Określa ona, w jaki sposób aplikacja Snake będzie budowana i uruchamiana. To właśnie w ramach tej konfiguracji są określane opcje emulatora oraz punkt wejścia aplikacji.

Należy osobno utworzyć konfigurację uruchomieniową oraz konfigurację do debugowania aplikacji. Każda z nich ma niezależne opcje. Oba rodzaje konfiguracji tworzy się przy użyciu opcji dostępnych w menu *Run* (*Run/Run Configuration* oraz *Run/Debug Configuration*).

Oto czynności, jakie należy wykonać, by stworzyć podstawową konfigurację uruchomieniową dla aplikacji Snake:

1. Wybrać opcję *Run/Run Configuration* (lub kliknąć projekt prawym przyciskiem myszy i z wyświetlonego menu kontekstowego wybrać opcję *Run As*).
2. Dwukrotnie kliknąć opcję *Android Application*.
3. Podać nazwę nowej konfiguracji, na przykład *SnakeRunConfiguration* (patrz rysunek 3.5).



Rysunek 3.5. Konfiguracja uruchomieniowa dla projektu Snake, zakładka Target z wybranym urządzeniem AVD

4. Wybrać projekt — kliknąć przycisk *Browse* i wybrać projekt *Snake*.
5. Przejść na zakładkę *Target* i z listy dostępnych AVD wybrać utworzone wcześniej urządzenie *Android_Vanilla2.2* (patrz rysunek 3.5).

Na zakładkach *Target* oraz *Common* można określać także inne opcje, jednak na razie pozostawimy w nich wartości domyślne.

Uruchamianie aplikacji Snake w emulatorze

Teraz możemy już uruchomić aplikację Snake; w tym celu wystarczy wykonać następujące czynności:

1. Rozwinąć menu widoczne przy ikonie *Run As* umieszczonej na pasku narzędzi (przedstawiającej zielone kółko z trójkącikiem).
2. Z menu wybrać utworzoną wcześniej konfigurację uruchomieniową *SnakeRunConfiguration*.
3. Począkać na uruchomienie emulatora Androida (co może chwilę potrwać).

Wskazówka

Należy się upewnić, że w momencie uruchamiania emulatora do komputera nie jest podłączone żadne urządzenie z Androidem (przy użyciu kabla USB). Domyślnym ustawieniem, zaznaczonym w sekcji *Deployment Target Selection Mode* (tryb wyboru urządzenia docelowego) na zakładce *Target* konfiguracji uruchomieniowej, jest wartość *Automatic*. Oznacza to, że jeśli do komputera będzie podłączone jakieś rzeczywiste urządzenie, to aplikacja Snake zostanie uruchomiona na nim, a nie w emulatorze.

4. Jeśli będzie to konieczne, przesunąć pasek na ekranie z lewa na prawo, by odblokować emulator (patrz rysunek 3.6).
5. Poczekać na uruchomienie aplikacji Snake, pokazanej na rysunku 3.7.

Emulator pozwala na interakcję z aplikacją Snake i faktyczne wypróbowanie tego, jak można się tą grą bawić. W przyszłości będzie ją można ponownie uruchomić w dowolnym momencie, korzystając z listy wszystkich aplikacji (w terminologii Androida nazywanej *App Drawer*).



Rysunek 3.6. Emulator w trakcie uruchamiania (zablokowany)

Budowanie pierwszej aplikacji na Androida

W końcu nadszedł czas, by napisać pierwszą aplikację na Androida. Zaczniemy od prostej aplikacji typu „Witaj, świecie”, którą następnie będziemy rozbudowywać, poznając różne możliwości platformy Android.

Wskazówka

Wiele przykładów przedstawionych w tym rozdziale to fragmenty aplikacji MyFirstAndroidApp. Jej kod źródłowy można znaleźć w przykładach dołączonych do książki, które można pobrać z serwera FTP wydawnictwa Helion, <ftp://ftp.helion.pl/przyklady/andrp2.zip>.




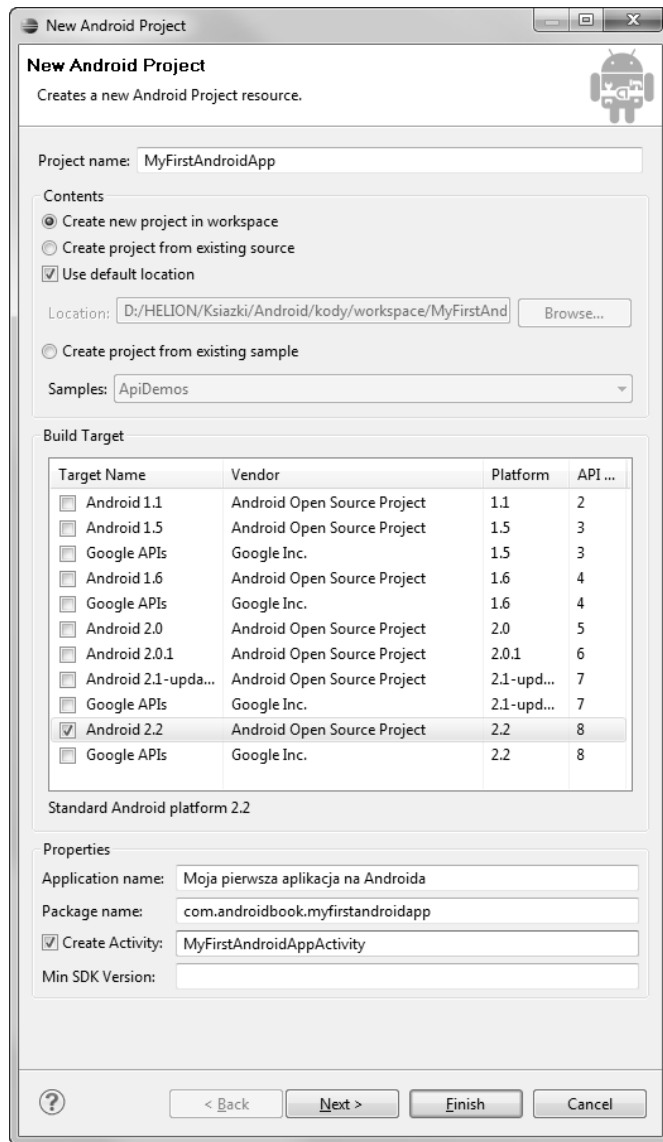
Rysunek 3.7. Gra Snake

Tworzenie i konfiguracja nowego projektu aplikacji

Sposób tworzenia nowego projektu aplikacji na Androida jest bardzo podobny do tego, który został opisany wcześniej przy okazji dodawania do obszaru roboczego Eclipse przykładowej aplikacji Snake.

Pierwszą czynnością, jaką należy wykonać, jest stworzenie nowego projektu i dodanie go do obszaru roboczego Eclipse. Kreator Android Project Wizard tworzy wszystkie pliki wymagane do utworzenia aplikacji na Androida. Poniżej opisane zostały wszystkie czynności, jakie należy wykonać, by utworzyć nowy projekt aplikacji na Androida w środowisku Eclipse:

1. Wybrać z menu opcję *File/New/Android Project* bądź kliknąć przycisk *Android Project* zawierający ikonę przedstawiającą folder z literą „a” oraz znakiem „+” () , umieszczony na pasku narzędzi Eclipse.
2. W polu *Project Name* podać nazwę tworzonego projektu. W naszym przypadku nadamy mu nazwę `MyFirstAndroidApp`.
3. Określić lokalizację plików projektu. Ponieważ mamy zamiar utworzyć nowy projekt, należy zaznaczyć przycisk opcji *Create new project in workspace*. Dodatkowo należy zaznaczyć pole wyboru *Use default location* lub wskazać katalog, w którym chcemy umieścić pliki źródłowe tworzonej aplikacji.
4. Z listy *Build Target* wybrać docelową wersję platformy Android, na której będzie działać tworzona aplikacja. Warto wybrać wersję, która jest zgodna z urządzeniem, jakie posiadamy. W tym przypadku wybierzemy platformę Android 2.2.
5. Określić nazwę aplikacji. Jest to nazwa „przyjazna dla użytkownika”, która będzie prezentowana na liście aplikacji w urządzeniu. W naszym przypadku tworzonej aplikacji nadamy nazwę *Moja pierwsza aplikacja na Androida*.
6. Określić nazwę pakietu. W tym przypadku należy podać nazwę określoną zgodnie ze standardowymi konwencjami nazewnictwa przestrzeni nazw pakietów stosowanymi w języku Java. Ponieważ wszystkie przykłady zamieszczone w tej książce są umieszczone w przestrzeni nazw `com.androidbook.*`, zastosujemy nazwę pakietu o następującej postaci: `com.androidbook.myfirstandroidapp`; choć oczywiście Czytelnik może podać inną, własną nazwę pakietu.
7. Zaznaczyć pole wyboru *Create Activity*. Zaznaczenie tej opcji poinformuje kreatora o tym, że należy wygenerować domyślną aktywność (ang. *activity*) pozwalającą na uruchomienie aplikacji. Nasza wersja tej klasy, dziedzicząca po `Activity`, będzie nosić nazwę `MyFirstAndroidAppActivity`.
Ustawienia nowego projektu zostały przedstawione na rysunku 3.8.
8. W końcu można kliknąć przycisk *Finish*.
9. Aby nasza przykładowa aplikacja była prawidłowo spolonizowana, należy jeszcze zmodyfikować wyświetlane przez nią komunikaty. W tym celu w panelu *Project Explorer* Eclipse należy rozwinąć węzeł projektu, a następnie kliknąć kolejne opcje: *res* oraz *values* i dwukrotnie kliknąć plik *strings.xml*. W wyświetlonym edytorze zasobów należy zaznaczyć opcję *hello* — spowoduje to wyświetlenie z prawej strony dwóch pól tekstowych. W dolnym z nich, *Value*, należy zmienić słowa: „Hello World” na: „Witaj, świecie”. W końcu należy zapisać plik zasobów, naciskając kombinację klawiszy `Ctrl+S`. Te ostatnie czynności mogą się wydać Czytelnikowi dosyć tajemnicze, jednak wszystko wyjaśni się w dalszej części książki, a w szczególności w rozdziałach 6. i 7., poświęconych odpowiednio zarządzaniu zasobami oraz wizualnym elementom interfejsu użytkownika aplikacji na Androida.



Rysunek 3.8. Konfiguracja tworzonej aplikacji przy użyciu narzędzia Android Project Wizard

Podstawowe pliki i katalogi aplikacji na Androida

Każda aplikacja przeznaczona na platformę Android posiada pewną grupę podstawowych plików tworzonych i używanych w celu określenia jej funkcjonalności (patrz tabela 3.1). Pliki te są tworzone domyślnie podczas generowania nowej aplikacji.

Tabela 3.1. Ważne pliki i katalogi projektów aplikacji na Androida

Plik	Ogólny opis
<i>AndroidManifest.xml</i>	Globalny plik opisujący aplikację. Definiuje on możliwości oraz uprawnienia aplikacji, jak również sposób jej działania.
<i>default.properties</i>	Tworzony automatycznie plik projektu. Określa on docelową wersję platformy Androida, na którą jest tworzona aplikacja, oraz inne opcje związane z jej budowaniem.
katalog <i>src</i>	Wymagany folder, w którym są przechowywane wszystkie kody źródłowe tworzonej aplikacji.
<i>src/com.androidbook.myfirstandroidapp/MyFirstAndroidAppActivity.java</i>	Główny plik źródłowy, definiujący punkt wejścia aplikacji.
folder <i>gen</i>	Wymagany folder, w którym są przechowywane automatycznie generowane pliki zasobów aplikacji.
<i>gen/com.androidbook.myfirstandroidapp/R.java</i>	Automatycznie wygenerowany plik źródłowy służący do zarządzania zasobami; nigdy nie powinien być edytowany ręcznie.
folder <i>res</i>	Wymagany folder, w którym są gromadzone wszystkie zasoby używane przez aplikację. Do tych zasobów zaliczamy: animacje, obrazy i inne pliki przeznaczone do wyświetlania, pliki układów, pliki XML, zasoby danych (takich jak łańcuchy znaków) oraz dowolne inne pliki.
<i>res/drawable-*/icon.png</i>	Foldery zasobów, w których są przechowywane ikony aplikacji dla urządzeń o różnych rozdzielczościach wyświetlaczy.
<i>res/layout/main.xml</i>	Pojedynczy plik układu.
<i>res/values/strings.xml</i>	Plik zasobów aplikacji zawierający łańcuchy znaków.
folder <i>assets</i>	Folder, w którym są przechowywane wszystkie „aktywa” aplikacji — czyli dane (zarówno pliki, jak i katalogi), które nie powinny być traktowane i zarządzane jako zasoby.

Oprócz tego w ramach tworzenia nowego projektu na dysku komputera jest także zapisywanych kilka innych plików. Niemniej jednak pliki wymienione w tabeli 3.1 są najważniejsze, a tworząc aplikacje, będziemy ich regularnie używali.

Tworzenie AVD na potrzeby projektu

Kolejnym krokiem jest utworzenie wirtualnego urządzenia, AVD, które w najlepszy sposób będzie opisywać urządzenie, jakie będziemy chcieli symulować podczas testowania i uruchamiania aplikacji. W tym przypadku wykorzystamy to samo AVD, które utworzyliśmy w celu uruchomienia przykładowej aplikacji Snake. AVD opisuje urządzenie, a nie aplikację. Dzięki temu tego samego AVD można używać podczas pisania i testowania wielu aplikacji. Istnieje także możliwość tworzenia podobnych

urządzeń wirtualnych, o tej samej konfiguracji, lecz zawierających różne dane (czyli różne zainstalowane aplikacje oraz różne dane na karcie pamięci).

Uwaga

Więcej informacji na temat tworzenia różnych rodzajów AVD oraz sposobów korzystania z emulatora Androida można znaleźć w dodatku A, „Krótki przewodnik po emulatorze Androida”.

Tworzenie konfiguracji uruchomieniowych dla projektu

Kolejną czynnością będzie utworzenie konfiguracji uruchomieniowej oraz konfiguracji testowej służącej do debugowania projektu. Określają one odpowiednio, w jaki sposób oraz w jakich warunkach aplikacja będzie uruchamiana i debugowana. W ramach konfiguracji uruchomieniowej określone są opcje emulatora Androida oraz punkt wejścia do aplikacji.

Oba rodzaje konfiguracji — uruchomieniowa oraz do debugowania — są tworzone niezależnie od siebie i posiadają inne opcje. Zacznijmy od utworzenia konfiguracji uruchomieniowej.

Poniższa lista przedstawia czynności, jakie należy wykonać, by stworzyć konfigurację uruchomieniową dla aplikacji `MyFirstAndroidApp`:

1. Z menu Eclipse wybrać opcję *Run/Run Configuration* (lub kliknąć projekt prawym przyciskiem myszy i z wyświetlonego menu kontekstowego wybrać opcję *Run As*).
2. Dwukrotnie kliknąć opcję *Android Application*.
3. Określić nazwę konfiguracji — `MyFirstAndroidRunConfig`.
4. Wybrać uruchamiany projekt — kliknąć przycisk *Browse* i w wyświetlonym oknie dialogowym wybrać projekt `MyFirstAndroidApp`.
5. Przejść na zakładkę *Target* i w sekcji *Default Target Selection Mode* zaznaczyć przycisk opcji *Manual*.
6. Kliknąć przycisk *Apply*, a następnie *Close*.

Wskazówka

Jeżeli podczas definiowania konfiguracji uruchomieniowej lub testowej w sekcji *Default Target Configuration Mode* zostanie zaznaczony przycisk opcji *Automatic*, to jeżeli tylko do komputera będzie podłączone urządzenie, aplikacja zostanie na nim zainstalowana i uruchomiona. W przeciwnym razie aplikacja zostanie zainstalowana i uruchomiona na emulatorze, na wskazanym wirtualnym urządzeniu AVD. Jeżeli jednak zostanie zaznaczony przycisk opcji *Manual*, a nie *Automatic*, to Eclipse zawsze będzie nas pytać, czy: a) chcemy uruchomić aplikację w istniejącym emulatorze; b) chcemy uruchomić aplikację w nowym emulatorze i wskazać AVD, jakiego należy przy tym użyć; c) chcemy uruchomić aplikację na fizycznym urządzeniu (jeżeli jest ono podłączone do komputera). Jeżeli jakiś emulator już działa, a następnie zostanie podłączone urządzenie, to jeśli w konfiguracji uruchomieniowej został zaznaczony przycisk opcji *Automatic*, opisane opcje wyboru także zostaną wyświetlone.


Kolejną czynnością będzie utworzenie konfiguracji testowej, umożliwiającej debugowanie aplikacji. Proces ten jest bardzo podobny do opisanego powyżej. Oto lista czynności, jakie należy wykonać, by stworzyć konfigurację testową dla aplikacji `MyFirstAndroidApp`:

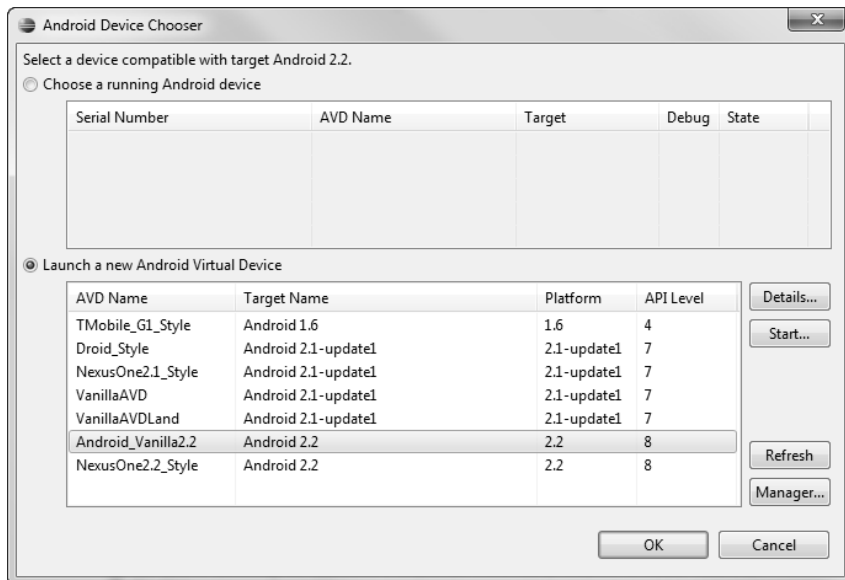
1. Z menu Eclipse wybrać opcję *Run/Debug Configuration* (lub kliknąć projekt prawym przyciskiem myszy i z wyświetlonego menu kontekstowego wybrać opcję *Debug As*).
2. Dwukrotnie kliknąć opcję *Android Application*.
3. Podać nazwę tworzonej konfiguracji — `MyFirstAndroidAppDebugConfig`.
4. Wybrać uruchamiany projekt — kliknąć przycisk *Browse* i w wyświetlonym oknie dialogowym wybrać projekt `MyFirstAndroidApp`.
5. Przejść na zakładkę *Target* i w sekcji *Default Target Selection Mode* zaznaczyć przycisk opcji *Manual*.
6. Kliknąć przycisk *Apply*, a następnie *Close*.

Teraz dysponujemy już konfiguracją, która pozwoli na debugowanie tworzonej aplikacji.

Uruchamianie aplikacji w emulatorze

Teraz możemy już uruchomić aplikację `MyFirstAndroidApp`; oto, co należy w tym celu zrobić:

1. Kliknąć strzałkę umieszczoną z prawej strony przycisku *Run As* znajdującego się na pasku narzędzi Eclipse (przedstawiającego zielone kółko z białym trójkącikiem) — .
2. Z wyświetlonego menu wybrać utworzoną wcześniej konfigurację. (Jeśli nie będzie ona widoczna na liście, to należy wybrać umieszczoną w dolnej części menu opcję *Run Configuration...* i wskazać odpowiednią konfigurację w wyświetlonym oknie dialogowym. W tym przypadku wybrana konfiguracja pojawi się następnym razem w menu).
3. Ponieważ w konfiguracji wybraliśmy ręczny (*Manual*) tryb wyboru urządzenia, teraz zostaniemy poproszeni o wskazanie, na jakim urządzeniu aplikacja ma zostać uruchomiona. W wyświetlonym oknie dialogowym należy zaznaczyć przycisk opcji *Launch a new Android Virtual Device*, a następnie zaznaczyć wirtualne urządzenie, które chcemy uruchomić (patrz rysunek 3.9).
4. Po kliknięciu przycisku *OK* zostanie uruchomiony emulator Androida (może to trochę potrwać).

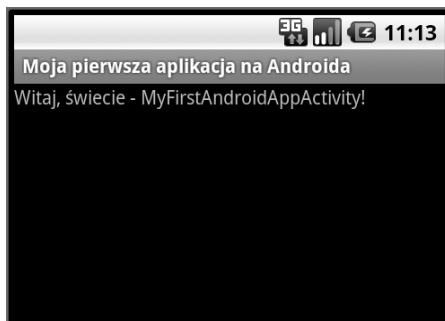


Rysunek 3.9. Ręczne określanie urządzenia, na jakim zostanie uruchomiona aplikacja

Wskazówka

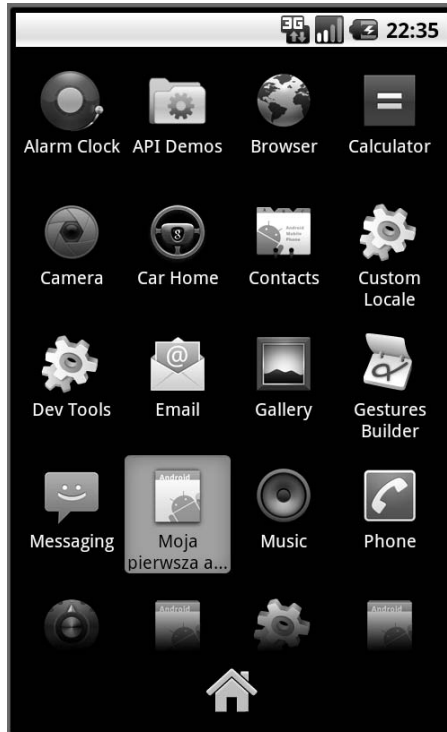
Uruchamianie emulatora może potrwać dosyć długo. Warto zatem pozostawić działający emulator i podłączać się do niego później, w razie takiej konieczności. Narzędzia dostępne w środowisku Eclipse obsługują ponowną instalację i uruchamianie aplikacji, dzięki czemu można zaoszczędzić czas i korzystać z wygody, jaką daje pozostawienie uruchomionego emulatora.

5. Kliknąć przycisk *Menu*, by odblokować emulator.
6. Bezpośrednio potem zostanie uruchomiona aplikacja, jak pokazano na rysunku 3.10.



Rysunek 3.10. Aplikacja Moja pierwsza aplikacja na Androida uruchomiona w emulatorze

7. Kliknąć przycisk *Home* w emulatorze, by zakończyć działanie aplikacji.
8. Wyświetlić listę zainstalowanych aplikacji, klikając ikonę listy aplikacji. Ekran emulatora powinien wyglądać podobnie do tego przedstawionego na rysunku 3.11.



Rysunek 3.11. Program *Moja pierwsza aplikacja na Androida* wyświetlony na liście aplikacji

9. Kliknąć ikonę *Moja pierwsza aplikacja na Androida*, aby ponownie uruchomić aplikację.

Debugowanie aplikacji w emulatorze

Zanim zajmiemy się kolejnymi zagadnieniami, Czytelnik koniecznie musi się dowiedzieć, jak można debugować aplikacje w emulatorze Androida. Aby mieć okazję do poznania i wypróbowania niektórych spośród dostępnych narzędzi do debugowania, celowo wprowadzimy błąd w kodzie naszej demonstracyjnej aplikacji.

W tym celu w Eclipse otworzymy plik *MyFirstAndroidApp.java* i dodajmy do niego nową metodę o nazwie *forceError()*. Wywołanie tej metody należy umieścić wewnątrz kodu metody *onCreate()*. Metoda *forceError()* spowoduje zgłoszenie nowego, nieprzechwytywanego wyjątku.

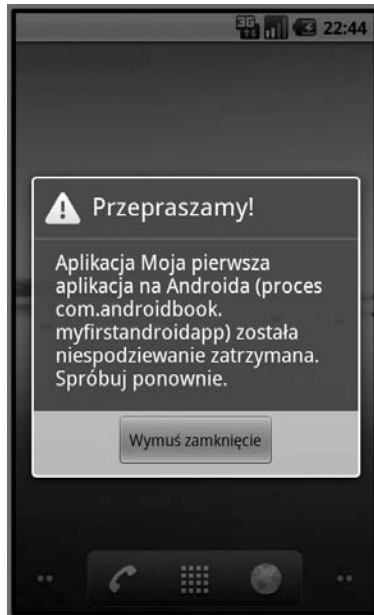
Poniżej przedstawiony został jej kod:

```
public void forceError() {
    if(true) {
        throw new Error("0 rany!");
    }
}
```

Wskazówka


Środowisko Eclipse zawiera osobne perspektywy (czyli grupy określonych paneli) służące do tworzenia kodu oraz debugowania aplikacji. Perspektywy można zmieniać, wybierając odpowiednie nazwy wyświetlone na wysokości paska narzędzi, z prawej strony okna środowiska. Perspektywa *Java* zawiera panele przeznaczone do tworzenia kodu oraz poruszania się po projekcie. Z kolei perspektywa *Debug* pozwala na definiowanie nowych punktów wstrzymania, przeglądanie informacji gromadzonych w dzienniku *LogCat* oraz debugowanie aplikacji. Kolejna perspektywa, *Dalvik Debug Monitor Service (DDMS)*, pozwala monitorować i modyfikować emulator oraz bieżący stan uruchomionego w nim wirtualnego urządzenia.

W tym momencie warto uruchomić aplikację i sprawdzić, co się stanie. W tym celu należy w pierwszej kolejności skorzystać z konfiguracji uruchomieniowej (*Run Configuration*). W emulatorze zobaczymy tylko tyle, że aplikacja została nieoczekiwanie zamknięta. Na ekranie emulatora zostanie wyświetlone okno dialogowe dające możliwość wymuszenia zamknięcia aplikacji (zostało ono przedstawione na rysunku 3.12).

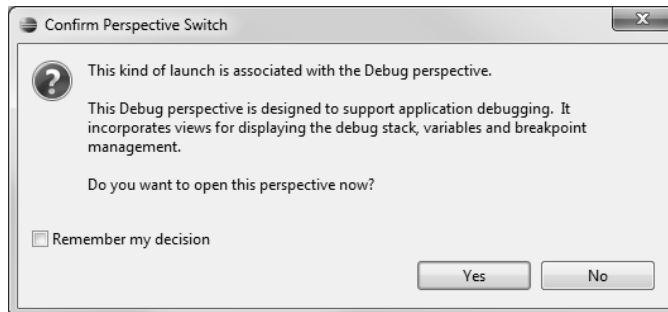


Rysunek 3.12. Aplikacja Moja pierwsza aplikacja na Androida zgłaszająca informacje o problemach

Teraz zamknijmy aplikację oraz emulator. Kiedy to zrobimy, będziemy mogli przystąpić do debugowania aplikacji. Można to zrobić, wykonując następujące czynności:

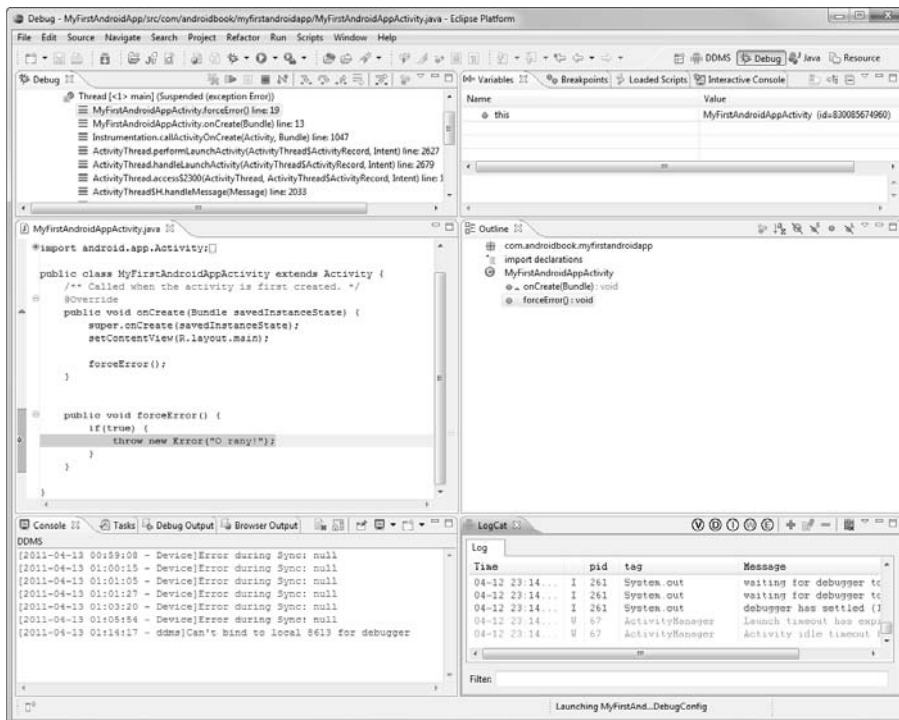
1. Kliknąć strzałkę widoczną z prawej strony przycisku *Debug As* umieszczonego na pasku narzędzi Eclipse (zawiera on ikonę zielonego robaka z czarną strzałką po prawej stronie) — .
2. Rozwinąć menu i wybrać z niego utworzoną wcześniej konfigurację testową. (Jeśli nie będzie ona widoczna na liście, należy wybrać umieszczoną w dolnej części menu opcję *Debug Configuration...* i wskazać odpowiednią konfigurację w wyświetlonym oknie dialogowym. W tym przypadku wybrana konfiguracja pojawi się następnym razem w menu).
3. Wykonać te same czynności, które zostały przedstawione w opisie sposobu uruchamiania aplikacji — wybrać odpowiednie AVD, poczekać na uruchomienie emulatora i w razie potrzeby go odblokować.

Uruchomienie emulatora oraz dołączenie do niego debugera może chwilę zająć. Jeśli Czytelnik będzie debugował aplikację na Androida po raz pierwszy, zostanie wyświetlonych kilka okien dialogowych, takich jak to przedstawione na rysunku 3.13 — należy je przejrzeć i zamknąć.



Rysunek 3.13. Zmiana perspektywy w celu debugowania aplikacji uruchomionej w emulatorze Androida

W środowisku Eclipse można skorzystać z perspektywy *Debug*, aby ustawiać punkty wstrzymania (ang. *breakpoint*), wykonywać program instrukcja po instrukcji oraz przeglądać informacje zarejestrowane w dzienniku *LogCat*. Tym razem gdy pojawią się problemy w aplikacji, korzystając z debugera, będzie można określić ich przyczynę. Być może w celu skonfigurowania debugera konieczne będzie przejrzanie i zamknięcie kilku okien dialogowych. Jeśli zezwolimy na dalsze wykonywanie aplikacji po zgłoszeniu wyjątku, to wyniki jej działania zostaną wyświetlone w perspektywie *Debug*. Przeglądając informacje wyświetlone w panelu *LogCat*, dowiemy się, że aplikacja została zamknięta z powodu nieobsłużonego wyjątku (patrz rysunek 3.14).



Rysunek 3.14. Debugowanie przykładowej aplikacji Moja pierwsza aplikacja na Androida

Konkretnie rzecz biorąc, na czerwono został wyświetlony błąd `AndroidRuntime: java.lang.Error: 0 rany!`.

Wróćmy do emulatora, klikając przycisk *Force Close*. Spróbujmy ustawić punkt wstrzymania na metodzie `forceError()`. W tym celu wystarczy kliknąć wybrany wiersz kodu prawym przyciskiem myszy i z wyświetlonego menu kontekstowego wybrać opcję *Toggle Breakpoint* (alternatywną metodą jest naciśnięcie kombinacji klawiszy `Ctrl+Shift+B`).

Wskazówka

W środowisku Eclipse można krokowo wykonywać kolejne instrukcje, używając poleceń *Step Into* (F5), *Step Over* (F6), *Step Return* (F7) oraz *Resume* (F8).

W systemie Mac OS X może się okazać, że klawisz F8 jest globalnie skojarzony z pewnymi operacjami. W takim przypadku, aby w Eclipse skorzystać z możliwości ułatwionego wykonywania czynności poprzez naciśnięcie odpowiedniego klawisza, trzeba będzie zmienić skojarzenia klawiszy z operacjami. W tym celu należy wybrać z menu opcję *Windows/Preferences/General/Keys*, a następnie odnaleźć polecenie, z jakim jest skojarzony klawisz F8, i je zmienić. Alternatywnym rozwiązaniem będzie zmiana globalnego skojarzenia wykorzystującego klawisz F8. Aby to zrobić w systemie Mac OS X, należy wyświetlić *System Preferences/Keyboard & Mouse/Keyboard Shortcuts*, a następnie zmienić operację skojarzoną z klawiszem F8.

Teraz powinniśmy wrócić do emulatora, ponownie uruchomić w nim naszą przykładową aplikację i zacząć wykonywać ją instrukcja po instrukcji. Przekonamy się, że aplikacja zgłasza wyjątek, który następnie zostanie wyświetlony w panelu *Variable Browser* wchodzącym w skład perspektywy *Debug*. Rozwinięcie tej zmiennej pozwoli wyświetlić zawartość błędu "0 rany!".

To doskonały moment, by kilkakrotnie doprowadzić do awarii aplikacji i przyzwyczać się do przycisków i opcji pozwalających na sterowanie jej wykonywaniem. A skoro już zajmujemy się tymi zagadnieniami, wróćmy do perspektywy *DDMS*. Warto zwrócić uwagę, że emulator udostępnia listę procesów aktualnie działających na telefonie, takich jak system `_process` lub `com.android.phone`. Jeśli uruchomimy aplikację `MyFirstAndroidApp`, to na tej liście pojawi się opcja `com.androidbook.myfirstandroidapp`. Wystarczy teraz wymusić zamknięcie aplikacji, a odpowiadająca jej pozycja na liście zniknie. Perspektywy *DDMS* można używać do wymuszania zakończenia procesów, sprawdzania istniejących wątków oraz sterty, jak również do przeglądania systemu plików telefonu.

Dodawanie mechanizmów rejestracji do aplikacji

Zanim Czytelnik zacznie badać różne możliwości Android SDK, koniecznie powinien poznać możliwości rejestracji, jest to bowiem nieocenione narzędzie, które można wykorzystywać nie tylko do debugowania, lecz także poznawania Androida. Możliwości rejestracji zapewnia klasa `Log` wchodząca w skład pakietu `android.util`.

Kilka użytecznych metod tej klasy zostało przedstawionych w tabeli 3.2.

Tabela 3.2. *Użyteczne metody klasy android.util.Log*

Metoda	Przeznaczenie
<code>Log.e()</code>	Rejestruje błędy.
<code>Log.w()</code>	Rejestruje ostrzeżenia.
<code>Log.i()</code>	Rejestruje komunikaty informacyjne.
<code>Log.d()</code>	Rejestruje komunikaty do debugowania.
<code>Log.v()</code>	Rejestruje rozszerzone komunikaty informacyjne.

Wskazówka

Środowisko Eclipse stara się oszczędzać czas programistów — pozwala używać importowanych klas, a następnie dodać odpowiednie instrukcje importu automatycznie. Wystarczy wskazać nazwę klasy myszką i w wyświetlonym okienku dialogowym wybrać opcję *Import 'Log' (android.util)*.

Oprócz tego można także skorzystać z polecenia *Organize imports* (w systemie Windows można je wykonać, naciskając kombinację klawiszy *Ctrl+Shift+O*, a w Mac OS X *Command+Shift+O*), które sprawi, że Eclipse automatycznie doda wszelkie niezbędne instrukcje `import` dla używanych klas i pakietów oraz usunie te, które nie są już potrzebne. Jeśli pojawią się jakieś konflikty nazw, co w przypadku klas o nazwie `Log` zdarza się stosunkowo często, można określić pakiet, do którego należy importowana klasa.

Aby dodać możliwość rejestracji komunikatów do naszej aplikacji `MyFirstAndroidApp`, wystarczy otworzyć plik `MyFirstAndroidApp.java` w edytorze. W pierwszej kolejności na górze pliku należy dodać odpowiednią instrukcję `import`, która umożliwi korzystanie z klasy `Log`:

```
import android.util.Log;
```

Następnie w klasie `MyFirstAndroidApp` należy zadeklarować stałą łańcuchową, która będzie używana do oznaczania wszystkich rejestrowanych komunikatów. Później będziemy mogli skorzystać z możliwości, jakie daje panel `LogCat` Eclipse, by wyświetlać w nim tylko te komunikaty, które mają naszą etykietę:

```
private static final String DEBUG_TAG = "MyFirstAppLogging";
```

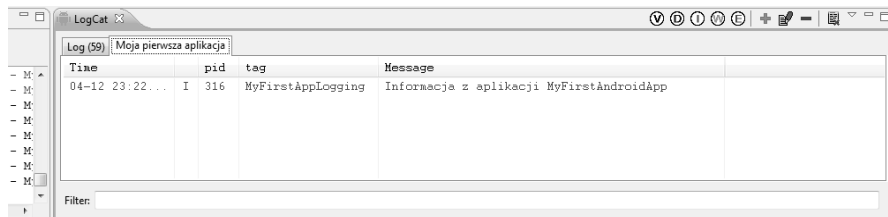
Teraz w kodzie metody `onCreate()` możemy już zarejestrować jakiś komunikat informacyjny:

```
Log.i(DEBUG_TAG, "Informacja o aplikacji MyFirstAndroidApp");
```

Ostrzeżenie

Skoro już wróciliśmy do kodu metody `onCreate()`, to Czytelnik musi pamiętać, by umieścić w komentarzu wywołanie metody `forceError()`, aby aplikacja ponownie zaczęła działać prawidłowo.

Teraz jesteśmy już gotowi do uruchomienia naszej aplikacji. A zatem zapiszmy zmiany wprowadzone w jej kodzie źródłowym i uruchommy ją w emulatorze, korzystając z konfiguracji do debugowania. Należy zwrócić uwagę, że na liście w panelu `LogCat` pojawił się nasz komunikat informacyjny, oznaczony etykietą `MyFirstAppLogging` wyświetlona w kolumnie `tag` (patrz rysunek 3.15).



Rysunek 3.15. Lista komunikatów `LogCat` przefiltrowana tak, by wyświetlać wyłącznie komunikaty z etykietą `MyFirstAppLogging`

Wskazówka

Czytelnik może uznać, że warto utworzyć w panelu `LogCat` specjalny filtr, który spowoduje, że będą w nim wyświetlane wyłącznie komunikaty posiadające jego etykietę. W tym celu należy kliknąć ikonę przedstawiającą zielony znak plusa, umieszczoną w prawej górnej części panelu `LogCat`. Następnie należy podać nazwę tworzonoego filtra, na przykład `MyFirstApp`, i wpisać wartość `MyFirstAppLogging` w polu `by Log Tag`. Teraz w panelu `LogCat` pojawiła się druga zakładka, prezentująca jedynie komunikaty wybrane przez nas.

Dodawanie obsługi multimediów do aplikacji

Kolejnym testem będzie próba dodania do naszej aplikacji niewielkich „wodotrysków” — konkretnie rzecz biorąc, spróbujemy odtworzyć w niej plik muzyczny zapisany w formacie MP3. Możliwości odtwarzania multimediów w Androidzie zostały zaimplementowane w klasie `MediaPlayer` należącej do pakietu `android.media`.

Obiekt `MediaPlayer` można utworzyć na podstawie istniejących zasobów lub określając adres pliku docelowego w postaci identyfikatora URI. W celu jak największego uproszczenia przykładu zaczniemy od wskazania pliku MP3 przy użyciu identyfikatora URI, czyli obiektu klasy `Uri` należącej do pakietu `android.net`.

Wybrane metody klas `android.media.MediaPlayer` oraz `android.net.Uri` zostały przedstawione w tabeli 3.3.

Tabela 3.3. Ważne metody klasy `MediaPlayer` oraz metody do przetwarzania URI

Metoda	Przeznaczenie
<code>MediaPlayer.create()</code>	Tworzy nowy odtwarzacz muzyczny, który będzie operował na wskazanym pliku.
<code>MediaPlayer.start()</code>	Rozpoczyna odtwarzanie pliku.
<code>MediaPlayer.stop()</code>	Zatrzymuje odtwarzanie.
<code>MediaPlayer.release()</code>	Zwalnia zasób używany przez dany obiekt odtwarzacza.
<code>Uri.parse()</code>	Tworzy obiekt <code>Uri</code> na podstawie prawidłowo sformułowanego adresu URI.

Aby dodać do naszej przykładowej aplikacji możliwość odtwarzania plików MP3, trzeba wprowadzić pewne modyfikacje w pliku `MyFirstAndroidApp.java`. W pierwszej kolejności należy dodać instrukcje `import`, które pozwolą korzystać z klasy `MediaPlayer`:

```
import android.media.MediaPlayer;
import android.net.Uri;
```

Następnie w kodzie klasy `MyFirstAndroidApp` należy zadeklarować zmienną, w której będzie przechowywany obiekt `MediaPlayer`:

```
private MediaPlayer mp;
```

Kolejną czynnością będzie dodanie metody o nazwie `playMusicFromWeb()` i wywołanie jej wewnątrz metody `onCreate()`. Metoda ta w pierwszej kolejności utworzy obiekt `Uri`, następnie obiekt `MediaPlayer`, a następnie rozpocznie odtwarzanie wskazanego pliku MP3. Jeśli z jakiegokolwiek powodu operacji nie uda się wykonać prawidłowo, to metoda ta zarejestruje w dzienniku odpowiedni komunikat opatrzonej naszą utworzoną wcześniej etykietą.

Poniżej przedstawiony został kod metody `playMusicFromWeb()`:

```
public void playMusicFromWeb() {
    try {
        Uri file = Uri.parse("http://www.perlgurl.org/podcast/archives/
        ↳podcasts/PerlGurlPromo.mp3");
        mp = MediaPlayer.create(this, file);
    }
}
```

```

        mp.start();
    }
    catch (Exception e) {
        Log.e(DEBUG_TAG, "Błąd odtwarzacza", e);
    }
}

```

I w końcu, w momencie zamykania aplikacji będziemy chcieli usunąć z pamięci wszystkie używane wcześniej obiekty. W tym celu należy przesłonić metodę `onStop()`, a wewnątrz niej zatrzymać odtwarzacz i usunąć z pamięci wszystkie jego zasoby.

Wskazówka

W środowisku Eclipse można kliknąć prawym przyciskiem myszy wewnątrz kodu klasy i z wyświetlonego menu kontekstowego wybrać opcję *Source* (alternatywnym rozwiązaniem jest naciśnięcie kombinacji klawiszy *Alt+Shift+S*). W kolejnym menu należy wybrać opcję *Override/Implement Methods*, a następnie w wyświetlonym oknie dialogowym zaznaczyć pole wyboru przy metodzie `onStop()` i kliknąć przycisk *OK*.

Nasza wersja metody `onStop()` powinna wyglądać w następujący sposób.

```

protected void onStop() {
    if (mp != null) {
        mp.stop();
        mp.release();
    }
    super.onStop();
}

```

Kiedy teraz uruchomimy aplikację `MyFirstAndroidApp` w emulatorze (i jeśli dodatkowo będziemy dysponowali połączeniem z internetem, by pobrać plik MP3 określony przez podany w kodzie identyfikator URI), aplikacja zacznie odtwarzać plik muzyczny. Po zamknięciu aplikacji odtwarzacz `MediaPlayer` jest zatrzymywany, a jego zasoby prawidłowo usuwane z pamięci.

Dodawanie do aplikacji usług lokalizacyjnych

Nasza aplikacja już wie, jak się przywitać, ale nie wie, gdzie się znajduje. A zatem nadarza się chyba dobra okazja, by zapoznać się z niektórymi podstawowymi możliwościami pobierania współrzędnych GPS.

Tworzenie AVD wyposażonego w interfejsy API firmy Google

Aby móc pobawić się nieco usługami lokalizacyjnymi oraz skorzystać z możliwości integracji aplikacji z mapami, konieczne będzie wykorzystanie niektórych aplikacji firmy Google dostępnych na telefonach z Androidem — konkretnie rzecz biorąc, chodzi o aplikację `Google Maps`. Aby móc z niej korzystać, musimy utworzyć następne wirtualne urządzenie. To nowe AVD powinno mieć dokładnie takie same parametry i ustawienia jak nasze pierwsze AVD, `Android_Vanilla2.2`, z jednym wyjątkiem — w jego przypadku z listy

Target należy wybrać odpowiednik interesującego nas poziomu API (którym jest API Level 8) przygotowany przez Google (czyli opcję *Google APIs (Google Inc.) — API Level 8*). Temu nowemu AVD możemy nadać nazwę na przykład *Android_with_GoogleAPIs_2.2*.

Konfiguracja lokalizacji w emulatorze

Po utworzeniu nowego AVD wyposażonego w obsługę interfejsów API firmy Google konieczne jest zamknięcie aktualnie działającego emulatora. Następnie ponownie należy rozpocząć debugowanie naszej testowej aplikacji, jednak tym razem wybierając nowe AVD.

Emulator nie jest wyposażony w GPS, więc konieczne jest ręczne podanie współrzędnych określających jego początkowe położenie. W tym celu należy uruchomić emulator w trybie debugowania, wykorzystując przy tym AVD obsługujące interfejsy API Google'a, a następnie wykonać następujące czynności:

W emulatorze:

1. Nacisnąć przycisk *Home*, by wrócić do głównego ekranu.
2. Otworzyć listę aplikacji i uruchomić aplikację *Maps*.
3. Kliknąć przycisk *Menu*.
4. Wybrać opcję *Moja lokalizacja* (wyglądającą jak celownik z kółkiem w środku).

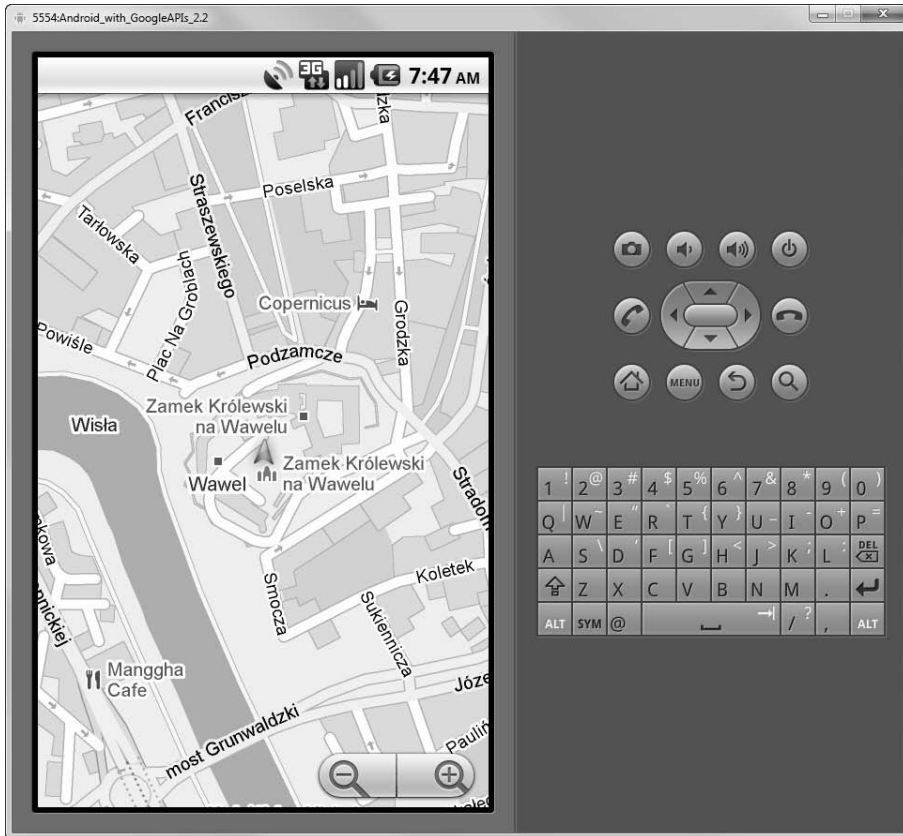
W Eclipse:

1. Kliknąć perspektywę *DDMS* umieszczoną w prawym górnym wierzchołku okna.
2. Z lewej strony okna Eclipse widoczny będzie panel *Emulator Control*. Należy go przewinąć w dół, tak by pojawiła się sekcja *Location Controls*.
3. W polach *Longitude* i *Latitude* wpisać odpowiednio długość i szerokość geograficzną. (Warto zwrócić uwagę, że ich kolejność jest zamieniona w stosunku do zwyczajowego sposobu podawania współrzędnych geograficznych).
4. Kliknąć przycisk *Send*.

Wskazówka

Aby określić konkretne współrzędne geograficzne, można wyświetlić w przeglądarce stronę <http://maps.google.pl>, po czym odnaleźć interesujące nas miejsce, kliknąć je prawym przyciskiem myszy i wybrać opcję *Co tu jest* z menu kontekstowego. Współrzędne wybranego miejsca pojawią się w polu tekstowym nad mapą. Na przykład odszukajmy Wzgórze Wawelskie, a po jego kliknięciu w polu tekstowym nad mapą zostaną wyświetlone współrzędne: *50,054005,19,935154*, czyli szerokość geograficzna wynosi *50,054005*, a długość *19,935154*.

Jeśli teraz wrócimy do emulatora, zauważymy, że aplikacja Google Maps pokazuje lokalizację, której współrzędne podaliśmy w Eclipse. Na ekranie telefonu w emulatorze powinna być widoczna mapa Krakowa, pokazana na rysunku 3.16.



Rysunek 3.16. Określanie położenia Wzgórza Wawelskiego w emulatorze

Jak widać, emulator wskazuje teraz symulowane położenie.

Odnajdowanie ostatniego znanego położenia

Aby dodać obsługę lokalizacji do naszej demonstracyjnej aplikacji `MyFirstAndroidApp`, trzeba będzie wprowadzić kilka zmian w pliku `MyFirstAndroidApp.java`. Przede wszystkim konieczne będzie dodanie odpowiednich instrukcji `import`:

```
import android.location.Location;
import android.location.LocationManager;
```

Następnie należy zadeklarować w klasie nową metodę — `getLocation()` — i wywołać ją w metodzie `onCreate()`. Metoda będzie pobierać ostatnią znaną lokalizację telefonu i wyświetlać ją w formie komunikatu zapisywanego w dzienniku. Jeśli z jakichkolwiek powodów operacji nie uda się wykonać prawidłowo, w dzienniku zostanie zapisana informacja o błędzie.

Poniżej pokazano, jak powinien wyglądać kod metody `getLocation()`:

```
public void getLocation() {
    try {
        LocationManager locMgr = (LocationManager)
            getSystemService(LOCATION_SERVICE);
        Location recentLoc = locMgr.
            getLastKnownLocation(LocationManager.GPS_PROVIDER);
        Log.i(DEBUG_TAG, "lokalizacja: " + recentLoc.toString());
    }
    catch (Exception e) {
        Log.e(DEBUG_TAG, "Nie udało się określić położenia", e);
    }
}
```

Dodatkowo nasza aplikacja wymaga odpowiednich uprawnień, by móc korzystać z możliwości określania położenia geograficznego. Uprawnienia te należy zarejestrować w pliku *AndroidManifest.xml*. Oto, co należy zrobić, by dodać do aplikacji uprawnienia pozwalające na korzystanie z usług lokalizacyjnych:

1. Dwukrotnie kliknąć plik *AndroidManifest.xml*.
2. Przejść na zakładkę *Permissions*.
3. Kliknąć przycisk *Add* i wybrać opcję *Uses Permission*.
4. W prawym panelu wybrać opcję `android.permission.ACCESS_FINE_LOCATION`.
5. Zapisać plik.

Jeśli teraz uruchomimy naszą przykładową aplikację w emulatorze, zarejestruje ona w dzienniku komunikat ze współrzędnymi, które przesłaliśmy wcześniej do emulatora; jego treść będzie można przeczytać w panelu *LogCat* w Eclipse.

Debugowanie aplikacji na fizycznym urządzeniu

Czytelnik opanował już sposoby uruchamiania aplikacji przy wykorzystaniu emulatora. Nadszedł zatem czas, by spróbować uruchomić ją na prawdziwym telefonie. W tym celu w pierwszej kolejności trzeba będzie oznaczyć aplikację jako przeznaczoną do debugowania. Stosowne informacje należy podać w pliku *AndroidManifest.xml*. Aby wprowadzić te zmiany, należy wykonać następujące operacje:

1. Dwukrotnie kliknąć plik *AndroidManifest.xml*.
2. Przejść na zakładkę *Application*.
3. Atrybutowi *Debuggable* przypisać wartość *True*.
4. Zapisać plik.

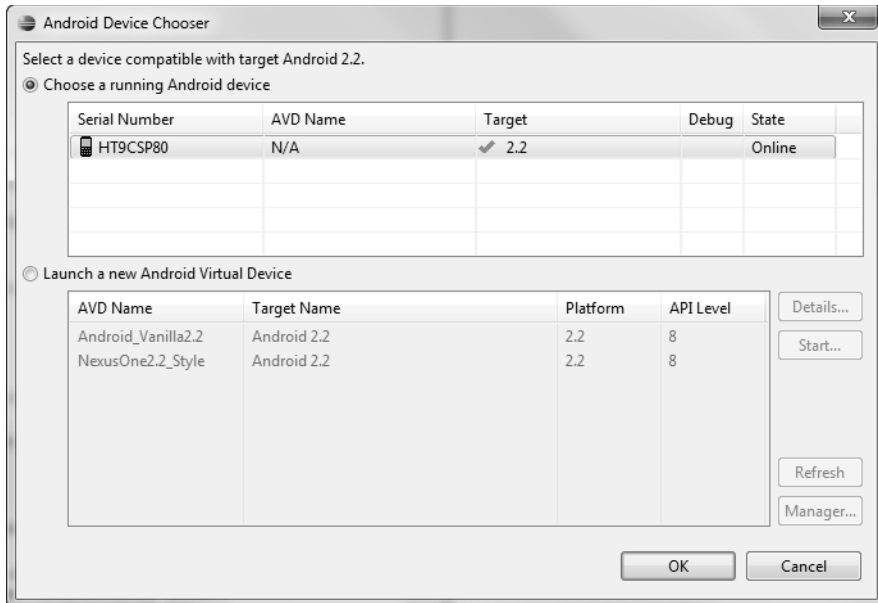
Innym rozwiązaniem jest wprowadzenie analogicznych zmian bezpośrednio w kodzie pliku *AndroidManifest.xml*, a konkretnie w atrybucie `android:debuggable`:

```
<application ... android:debuggable="true">
```

Jeśli zapomnimy przypisać atrybutowi `debuggable` wartości `true`, telefon będzie pokazywać okienko dialogowe z komunikatem o oczekiwaniu na podłączenie debugera

aż do momentu naciśnięcia przycisku *Force Close* i wprowadzenia odpowiednich zmian w pliku manifestu aplikacji.

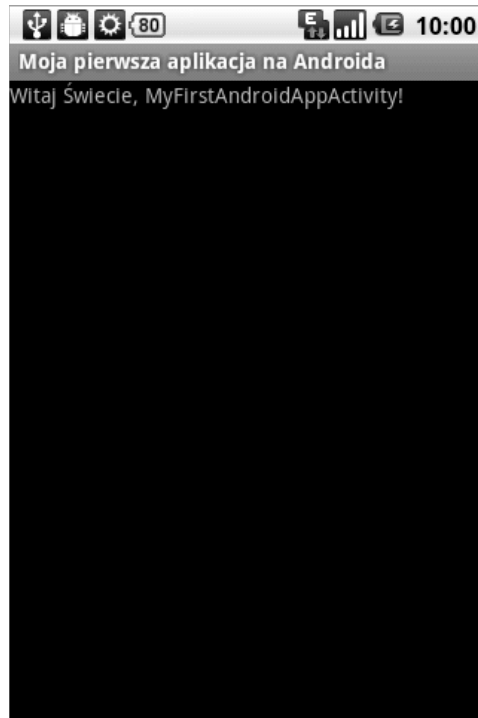
Teraz można już podłączyć telefon do komputera przy użyciu kabla USB i spróbować ponownie uruchomić aplikację (korzystając z konfiguracji uruchomieniowej lub testowej). Ponieważ w obu konfiguracjach ustawiliśmy ręczny tryb wyboru urządzenia, powinniśmy zobaczyć podłączone urządzenie jako opcję na liście działających urządzeń — *Choose a running Android device* (widocznej na rysunku 3.17).



Rysunek 3.17. Lista umożliwiająca wybór urządzenia, prezentująca telefon podłączony do komputera przy użyciu kabla USB

Teraz wystarczy wybrać telefon jako docelowe urządzenie, na którym chcemy uruchomić aplikację. W efekcie aplikacja zostanie skopiowana na telefon i uruchomiona tak samo jak wcześniej na emulatorze. Zakładając, że wcześniej zezwoliliśmy także na debugowanie aplikacji w telefonie, będziemy mogli analizować jej działanie tak samo jak na emulatorze. O tym, że telefon aktywnie używa połączenia w trybie debugowania, można się przekonać po ikonie „androidowego robaka” (🐛) wyświetlonej na pasku powiadomień telefonu. Rysunek 3.18 przedstawia zrzut aplikacji uruchomionej na prawdziwym telefonie.

Debugowanie aplikacji na telefonie odbywa się dokładnie tak samo jak w emulatorze, choć różni się od niego pod kilkoma względami. Nie można korzystać z kontrolki emulatora, by przesyłać do telefonu na przykład SMS-y lub informacje o położeniu geograficznym; można natomiast wykonywać analogiczne czynności — czyli napisać SMS lub włączyć odbiornik GPS, by określić położenie.



Rysunek 3.18. Aplikacja MyFirstAndroidApp uruchomiona na rzeczywistym telefonie

Podsumowanie

W tym rozdziale pokazano, jak można tworzyć, budować, uruchamiać i debugować aplikacje na Androida przy wykorzystaniu środowiska Eclipse. Zaczęliśmy od przetestowania środowiska programistycznego, wykorzystując do tego aplikację przykładową dostarczoną wraz z Android SDK, a następnie stworzyliśmy zupełnie nową aplikację. Czytelnik dowiedział się także, jak wprowadzać proste modyfikacje w aplikacji, i zobaczył kilka fascynujących możliwości Androida, które zostaną bardziej szczegółowo opisane w dalszej części książki.

W kilku kolejnych rozdziałach przedstawione zostaną bardziej szczegółowe informacje na temat definiowania możliwości i działania aplikacji przy użyciu pliku manifestu oraz opisany cykl działania aplikacji. Oprócz tego znajdą się w nich również zagadnienia związane z zarządzaniem zasobami, które będą używane w aplikacji, takimi jak obrazy oraz łańcuchy znaków.

Zasoby i inne źródła informacji

Sterowniki USB dla systemu Windows:

<http://developer.android.com/sdk/win-usb.html>

Poradnik dla programistów aplikacji na Androida:

<http://developer.android.com/guide/developing/device.html>

Skorowidz

<action>, 553
<activity>, 553, 561, 569
<alpha>, 148
<animation-list>, 148
<application>, 142
<array>, 148
<bool>, 147
<category>, 553
<color>, 147
<dimen>, 147
<drawable>, 148
<instrumentation>, 142
<integer>, 147
<integer-array>,, 148
<intent-filter>, 142, 553
<item>, 147, 148
<menu>, 148
<meta-data>, 561
<plurals>, 147, 159
<receiver>, 142, 542
<resources>, 149
<rotate>, 148
<scale>, 148
<selector>, 147
<service>, 520
<set>, 148
<string>, 147
<string-array>, 147
<style>, 148, 178, 181
<supports-screen>, 639
<TextView>, 189
<translate>, 148
<uses-configuration>, 135, 136, 607, 639
<uses-feature>, 136, 443, 468, 483, 639
<uses-sdk>, 133, 134, 468, 483, 548, 639

A

aapt, 146, 147
AbstractAccountAuthenticator, 574
AbstractThreadedSyncAdapter, 575
AccelerateDecelerateInterpolator, 293
AccelerateInterpolator, 293
ACCESS_COARSE_LOCATION, 428
Account, 574
account authenticator, 574
account providers, 574
AccountManager, 574
ACTION_DISCOVERY_FINISHED, 496
ACTION_DISCOVERY_STARTED, 496
ACTION_FOUND, 496
Activity, 109, 110, 460
adapter synchronizacji, 575
AdapterView, 256
ADB, 689
addOnPreDrawListener(), 221
ADV, 390
agent archiwizacji, 577, 580
AIDL, 519, 527
AIR, 384
akcja
 ACTION_BATTERY_CHANGED, 122
 ACTION_BATTERY_LOW, 122
aktualizacja aplikacji w Android Market, 723
aktywne foldery, 565, 566
aktywności, 109, 110
 cykl życia, 114
 stos, 112, 113
przechodzenie pomiędzy
 aktywnościami, 118

- AlarmManager, 538, 575
- AnalogClock, 214
- AndAppStore, 726
- Android, 46, 47, 48, 49, 54
 - architektura, 55
 - Cupcake, 50
 - Donut, 50
 - Éclair, 50
 - emulator, 70, 71, 85
 - Froyo, 50
 - Gingerbread, 50
 - glEsVersion, 443
 - logo, 49
 - maskotka, 49
 - nazwy kodowe, 50
 - prezentacja systemu, 35
 - szkielet aplikacji, 59
- Android 1.0 SDK, 133
- Android 1.1 SDK, 133
- Android 1.5 SDK (Cupcake), 133
- Android 1.6 SDK (Donut), 133
- Android 2.0 SDK (Éclair), 133
- Android 2.0.1 SDK (Éclair), 133
- Android 2.1 SDK (Éclair), 133
- Android 2.2 SDK, 133, 134
- Android Backup Service, 576, 577
- Android Cloud to Device Messaging (C2DM), 520
- Android Content Guidelines, 722
- Android Debug Bridge (ADB), 72
- Android Dev Phone, 659
- Android Developer Challenge (ADC), 46
- Android Development Tools (ADT), 61
- Android Device Chooser, 104
- Android Hierarchy Viewer, 73
- Android Interface Definition Language, 519
- Android Market, 48, 53, 57, 669, 712, 714, 717
- Android Native Development Kit, 442,
 - Patrz* Android NDK
- Android NDK, 477
 - instalacja, 476
 - stosowanie, 475
- Android Project, 78
- Android Project Wizard, 86, 88
- Android SDK, 58, 61, 64, 68, 306, 689
 - aktualizacja, 64
 - android.*, 68
 - dokumentacja, 66, 67
 - licencja, 65
 - narzędzia, 69
 - określanie wersji, 640
 - problemy, 64
- Android SDK (Gingerbread), 133
- Android SDK and AVD Manager, 69, 70
- Android SDK dalvik.*, 68
- Android SDK java.*, 68
- Android SDK javax.*, 68
- Android SDK junit.*, 68
- Android SDK License Agreement, 65
- Android SDK org.apache.http.*, 68
- Android SDK org.json, 68
- Android SDK org.w3c.dom, 68
- Android SDK org.xml.sax.*, 68
- Android SDK org.xmlpull.*, 68
- Android Virtual Device, 390
- android.accessibilityservice, 590
- android.accounts, 574
- android.bluetooth, 494
- android.content.Intent, 118
- android.gesture, 597
- android.hardware.Camera, 410
- android.hardware.Sensor, 487
- android.hardware.SensorManager, 486
- android.inputmethodservice, 588
- android.opengl, 442
- android.opengl.GLES11, 442
- android.opengl.GLES20, 442
- android.os.BatteryManager, 499
- android.permission.CALL_PHONE, 437
- android.permission.CAMERA, 419
- android.provider.CallLog, 439
- android.provider.LiveFolders, 567
- android.provider.Settings, 594
- android.provider.Telephony.SMS_RECEIVED, 436
- android.sax.*, 304
- android.speech, 590
- android.speech.tts, 593
- android.telephony, 428

- android.test.ActivityUnitTestCase, 687
- android.test.MoreAsserts, 687
- android.test.PerformanceTestCase, 687
- android.test.ServiceTestCase, 687
- android.test.TouchUtils, 687
- android.test.ViewAsserts, 687
- android.util.Xml.*, 304
- android.view.accessibility, 590
- android.view.inputmethod, 588
- android.webkit, 378
- android:backupAgent, 580
- android:collapseColumns, 253
- android:foreground, 245
- android:foregroundGravity, 245
- android:gravity, 248, 249
- android:icon, 569
- android:label, 569
- android:layout_above, 250
- android:layout_alignBottom, 250
- android:layout_alignLeft, 250
- android:layout_alignParentBottom, 250
- android:layout_alignParentLeft, 250
- android:layout_alignParentRight, 250
- android:layout_alignParentTop, 250
- android:layout_alignRight, 250
- android:layout_alignTop, 250
- android:layout_below, 250
- android:layout_centerHorizontal, 250
- android:layout_centerInParent, 249
- android:layout_centerVertical, 250
- android:layout_column, 253
- android:layout_gravity, 246, 248
- android:layout_height, 244
- android:layout_margin, 244
- android:layout_span, 253
- android:layout_toLeftOf, 250
- android:layout_toRightOf, 250
- android:layout_weight, 248
- android:layout_width, 244
- android:measureAllChildren, 246
- android:orientation, 248
- android:shrinkColumns, 253
- android:stretchColumns, 253
- AndroidManifest.xml, 118
- animacje, 285
 - poklatkowe, 285, 286
 - przejsć, 285, 288, 291
 - wczytywanie, 290
- AnimationDrawable, 286
- AnimationSet, 289
- animowane obrazy GIF, 285
- animowane tapety, 545
 - tworzenie, 545
 - instalacja, 549
 - konfiguracja, 547
- AnticipateInterpolator, 293
- AnticipateOvershootInterpolator, 293
- Antyaliasing, 269
- ApiDemos, 74
- aplikacja
 - aktualizacja w Android Market, 723
 - animowane tapety, 534
 - certyfikacja na Androida, 716
 - darmowe, 679
 - debugowanie, 62, 93, 103
 - dodawanie usług lokalizacyjnych, 100
 - dystrybucja, 717
 - filtry intencji, 534
 - ikona, 711
 - klient-serwer, 661
 - kontrola jakości, 662
 - LBS, 47
 - mechanizmy rejestracji, 97
 - mobilne
 - projektowanie, 664
 - architektura, 664
 - nazwa, 711
 - numer wersji, 712
 - określanie wymagań systemowych, 125
 - pielęgnacja oprogramowania, 669
 - polityka zwracania, 723
 - preferencje, 111
 - projektowanie pod kątem pielęgnacji, 665
 - projektowanie pod kątem współdziałania aplikacji, 667
 - projektowanie rozszerzeń, 665
 - przesyłanie do Android Market, 720
 - publikacja w Android Market, 722

aplikacja

- punkty integracji, 702
- sprzedawanie, 717, 725
- szkielet programowania, 68
- testowanie
 - aktualizacji oprogramowania
 - układowego, 670
 - aplikacji na urządzenia przenośne, 668
 - na urządzeniach, 660
 - publikowanej wersji pakietu, 716
- tworzenie na urządzenia przenośne, 667
- tworzenie, 648
- umiędzynarodawianie, 632, 637
- uruchamianie aktywności
 - udostępnianych przez aplikację, 125
- usuwanie z Android Market, 723
- wdrażanie, 668
- weryfikacja uprawnień, 713
- widzety, 533, 534
- wsparcie, 669
- wymagające jednokrotnej płatności, 679
- wymagające wykupienia subskrypcji, 679
- wymagania, 649
- wyświetlanie szczegółowych informacji
 - na temat aplikacji, 125
- wzbogacanie, 533
- z wbudowanymi opłatami, 679
- zarządzanie uprawnieniami aplikacji, 126
- zwracanie, 723
- AppWidgetProvider, 538
- ArcShape, 282
- ArrayAdapter, 256
- AsyncTask, 362
- Audio.Albums, 328
- Audio.Artists, 328
- Audio.Genres, 328
- Audio.Media, 328
- Audio.Playlists, 328
- AutoCompleteTextView, 196
- autoLink, 192
 - all, 192
 - email, 192
 - map, 192
 - none, 192
 - phone, 192
 - web, 192

AVD, 82

- tworzenie, 82, 89
- awarie
 - śledzenie informacji, 669
 - weryfikacja informacji, 669

B

- backup agent, 577
- BackupAgentHelper, 577
- bateria, monitorowanie stanu, 497
- baza danych, 306
 - firm trzecich, 656
- baza danych urządzeń, 654
 - korzystanie, 655
 - zarządzanie, 653
- biblioteka EGL, 448
- biblioteka GL, 450
- biblioteka OpenGL ES, 451
- biblioteki zewnętrzne, 137
- Bluetooth, 494
- BluetoothAdapter, 494, 495
- BluetoothDevice, 494
- BluetoothServerSocket, 494, 496
- BluetoothSocket, 494
- Bodlaender Hans, 273
- BounceInterpolator, 293
- BroadcastReceiver, 435, 436
- Browser, 328, 331
- bufor kolorów, 453
- Button, 200, 201
- ByteBuffer, 452

C

- CacheManager, 378
- CallLog, 328, 329
- CallLog.Calls, 439
- Camera, 410, 412
- CameraSurfaceView, 410, 412
- Canvas, 267, 269
- cena aplikacji, 721
 - darmowa, 721
 - Free, 721
 - Paid, 721
 - płatna, 721

certyfikacja aplikacji na Androida, 716
 certyfikacja oprogramowania, 662
 CheckBox, 200, 201
 Chronometer, 213
 cieniowanie pikseli, 467
 cieniowanie wierzchołkowe, 467
 clipping plane, 450
 completionThreshold, 197
 ConsoleMessage, 378
 ContactsContract, 328
 container, 255
 ContentResolver, 337, 424
 Context, 109, 418
 Context.deleteFile(), 302
 Context.listFiles(), 302
 Context.getCacheDir(), 302
 Context.getDir(), 302
 Context.getFilesDir(), 302
 Context.openFileInput(), 302
 Context.openFileOutput(), 302
 ContextMenu, 217
 CookieManager, 378
 Cursor, 313
 CursorAdapter, 256
 CustomGL2SurfaceView, 468
 CustomRenderer, 469
 CVS, 663
 CycleInterpolator, 293
 Cygwin 1.7, 476
 czas istnienia urządzeń na rynku, 657
 czerpanie dochodów z reklam, 727
 czujnik, 486

- kalibracja, 490
- odczyt danych, 488
- uzyskiwanie dostępu, 488

 czujnik Wi-Fi, 491

D

Dalvik Debug Monitor Server (DDMS), 71, 301
 dane, 145

- kojarzenie, 322

 dane użytkowników

- obsługa, 678
- przesyłanie, 678

DatePicker, 206, 207
 DDMS, 429, 682, 689
 debugowanie, 713
 DecelerateInterpolator, 293
 defekt, 692
 description, 548
 diagnostyka kodu, 686
 Dialog, 225, 226
 DigitalClock, 214
 DisplayMetrics, 617
 doAlert(), 382
 dochody z reklam, 727
 doConsoleLog(), 382
 dodawanie tekstur do obiektów, 457
 dokumentacja aplikacji, 670
 dokumentacja projektowa, 661, 662

- analiza wymagań, 661
- architektura aplikacji, 661
- dokumentacja zmian zakresu, 661
- określenie ryzyka, 661
- plany testowania aplikacji, 661
- pomiary wydajności, 661
- priorytety, 661
- projekt aplikacji, 661
- przypadki testowania, 661
- skrypty testowe, 661
- specyfikacja techniczna, 661
- szczegółowa specyfikacja interfejsu użytkownika, 661, 662
- zarządzanie ryzykiem, 661

 doSetFormText(), 382
 dostawcy danych CallLog, 329
 dostawcy kont, 574
 dostawcy treści, 139, 327, 341

- Browser, 328, 331
- CallLog, 328
- Contacts, 332
- ContactsContract, 328
- implementacja metod, 346
- MediaStore, 328
- modyfikacja danych, 336
- Settings, 328, 336
- UserDictionary, 328, 335

 doToast(), 381
 Draw Nine Patch, 682

Drawable, 278
 Droid Incredible, 607
 dystrybucja aplikacji, 717
 dzwonki, 425

E

Eclipse, 51, 61, 62, 94, 663, 689
 Eclipse Manifest File, 127, 128
 EditText, 193, 432, 437
 Edytory metod wprowadzania (IME), 588
 eglTerminate(), 464
 ekran, 625

- FWVGA, 625
- HVGA, 625
- menu głównego, 112
- najlepszych wyników, 112
- początkowy, 111
- QVGA, 625
- rozgrywki, 112
- systemu pomocy, 112
- VGA, 625
- WBGA, 625
- WQVGA, 625

 elementy konstrukcyjne aplikacji, 109

- aktywność, 109
- intencja, 109
- kontekst, 109

 EMS, 41
 emulator Androida, 689, 698
 Enhanced Messaging Service, Patrz EMS
 END_DOCUMENT, 361
 END_TAG, 361

F

FileBackupHelper, 578
 filtry intencji, 139

- rejestracja, 552

 Flash, 383
 Flash 10.1, 383
 Flash Lite, 383
 Flash Player for Android, 384
 Flash Professional CS5, 385
 FloatBuffer, 452, 457

format

- GIF, 241
- JPG, 241
- PNG, 241

 formatNumber(), 431
 fragment shader, 468
 fragmentacja, 613, 614
 FrameLayout, 235, 539, 616

G

GalleryRecord, 341
 GalleryView, 255
 GameAreaView, 598
 generowanie sugestii, 556
 Geocoder, 391
 GeoPoint, 401
 GestureDetector, 597, 598
 GestureDetector.SimpleOnGestureListener, 598
 GestureOverlayView, 597
 gesty, 596, 597, 604, 607
 getAuthToken(), 574
 GetMethodID(), 481
 GL_LINE_LOOP, 455
 GL_TEXTURE_COORD_ARRAY, 459
 GL_TRIANGLES, 455
 GL_VERTEX_ARRAY, 453
 GLDebugHelper, 449
 glDrawArrays(), 452, 453
 glDrawElements(), 452
 GLSurfaceView, 443, 444, 464, 465, 467
 GLSurfaceView.Renderer, 443, 464
 GNU Awk (Gawk), 476
 GNU Make 3.81, 476
 Google, 46, 47
 Google APIs, 390
 Google Nexus One, 659
 GPS, 388

- możliwości, 388
- określanie położenia, 388

 GpsSatellite, 406
 GpsStatus, 406
 GpsStatus.Listener, 406
 GPXService, 521, 522, 527

gradienty, 269
 grafika rastrowa, 275
 Graphics Interchange Format (GIF), 164
 gravity, 616
 GridView, 255, 567
 grupa klas, 235
 grupa uprawnień, 141

H

Handango, 725
 Handler, 460
 Hierarchy Viewer, 239, 240, 241, 682, 689
 Display View, 241
 Invalidate Layout, 242
 Pixel Perfect, 241, 242
 Request Layout, 242
 Tree View, 241
 HorizontalScrollView, 264
 HTC, 46
 HTTP, 358
 HttpURLConnection, 359

I

identyfikator URI, 341
 ikona, 131, 629, 711
 ImageButton, 203
 Images.Media, 328
 Images.ThumbNails, 328
 ImageUriAdapter, 343
 IME, 585
 implementacja oprogramowania
 mobilnego, 667
 includeInGlobalSearch, 563
 Input Method Editor, 585
 InputFilter, 198
 InputStream, 366
 inputText, 586
 insertOrThrow(), 309
 IntBuffer, 452
 Integrated Development Environment (IDE), 51
 intencja, 109, 121
 Intent, 109, 567

Intent.ACTION_ANSWER, 438
 Intent.ACTION_CALL, 438
 Intent.ACTION_DIAL, 438
 Intent.ACTION_VIEW, 438
 IntentFilter, 436
 interfejs użytkownika, 235
 organizacja, 238
 projektowanie, 615
 tworzenie, 233
 zakładki, 260
 Internet Assigned Numbers Authority
 (IANA), 551
 IRemoteInterface, 528
 ItemizedOverlay, 401, 402

J

Java Development Kit (JDK), 61
 java.xml.*, 304
 JavaScriptExtension, 380, 381
 javax.microedition.khronos.egl, 442
 javax.microedition.khronos.opengles, 442
 jednostki
 cale, 162
 milimetry, 162
 piksele, 162
 piksele niezależne od gęstości ekranu, 162
 piksele niezależne od skali, 162
 punkty, 162
 język XML, 360
 języki, 613
 Joint Photographics Expert Group (JPEG), 164

K

karta SD, 341, 342
 katalog aplikacji
 assets, 89
 gen, 89
 res, 89
 Khronos, 442
 KickBack, 590
 klawiatura programowa, 586
 niestandardowe, 588

- kolorowanie wierzchołków, 453
 - konfiguracja lokalizacji w emulatorze, 101
 - konfiguracja wyszukiwania, 555
 - konfiguracje sprzętowe, 613
 - kontekst aplikacji, 109, 110, 111
 - określanie uprawnień aplikacji, 111
 - pobieranie danych, 111
 - pobieranie zasobów zgromadzonych w aplikacji, 111
 - pobieranie, 110
 - sprawdzanie uprawnień aplikacji, 111
 - stosowanie, 110
 - uruchamianie aktywności, 111
 - zarządzanie bazami danych aplikacji, 111
 - zarządzanie katalogami aplikacji, 111
 - zarządzanie plikami aplikacji, 111
 - żądanie dostępu do usług systemowych, 111
 - konto programisty, 717, 724
 - kontrolka, 188
 - AnalogClock, 539
 - Button, 539
 - Chronometer, 539
 - ImageButton, 539
 - ImageView, 539
 - ProgressBar, 539
 - TextView, 539
 - kopie zapasowe, 576
 - kursory, 312
 - zarządzanie, 312
 - kwadrat roboczy, 621, 622
 - kwalifikatory katalogów zasobów alternatywnych, 624
 - dostępność klawiatury, 626
 - dostępność klawiszy nawigacyjnych, 627
 - dostępny sposób nawigacji, 627
 - gęstość pikseli na ekranie, 626
 - kod języka, 625
 - kod MCC, 625
 - kod MNC, 625
 - kod regionu, 625
 - metoda wprowadzania danych, 627
 - orientacja ekranu, 625
 - tryb dokowania, 625
 - tryb nocny, 626
 - typ ekranu dotykowego, 626
 - typ klawiatury, 626
 - używana platforma Android, 627
 - wielkość ekranu, 625
 - współczynnik proporcji ekranu, 625
- L**
- launchRecognizer, 558
 - launchWebSearch, 558
 - layout, 188
 - LayoutInflater, 343
 - liczby
 - całkowite, 298
 - całkowite typu long, 298
 - zmiennoprzecinkowe, 298
 - LinearGradient, 270
 - LinearInterpolator, 293
 - LinearLayout, 235, 539, 616
 - ListView, 255, 322, 616
 - Location, 391
 - LocationListener, 390, 524
 - LogCat, 98
 - lokalizacja treści, 340
 - LunarLander, 75
- Ł**
- łańcuchy znaków, 298
- M**
- maksymalizowanie zgodności aplikacji, 613
 - manipulator kulowy, 607
 - MapView, 396, 401
 - mash-up application, 47
 - Matrix, 276, 605
 - maxSdkVersion, 133, 135
 - mechanizm uwierzytelniający, 574
 - MediaController, 421
 - MediaPlayer, 423
 - MediaRecorder, 418, 421
 - MediaStore, 328
 - menedżer kont, 573

- menu
 - główne, 120
 - kontekstowe, 217
 - rozwijane, 120
- metoda
 - addJavaScriptInterface(), 381
 - addURI(), 568
 - animateTo(), 397
 - bindService(), 121, 528
 - BitmapFactory.decodeStream(), 366
 - boundCenterBottom(), 403
 - CallVoidMethod(), 482
 - cancelDiscovery(), 496
 - ContentUris.withAppendedId(), 334
 - Context.bindService(), 520, 522
 - Context.openFileInput(), 304
 - Context.startService(), 520, 522
 - dataChanged(), 581
 - delete(), 350
 - dismissDialog(), 225
 - distanceTo(), 390
 - doInBackground(), 363
 - doServiceStart(), 524
 - dostosowywania, 650, 651
 - draw(), 453
 - dystrybucji, 709
 - eglDestroyContext(), 464
 - eglDestroySurface(), 464
 - eglMakeCurrent(), 464
 - enable(), 495
 - ExceptionDescribe(), 482
 - ExceptionOccured(), 482
 - execSQL(), 308
 - filter(), 198
 - findViewById(), 421
 - formatNumber(), 431
 - getAccountByType(), 574
 - getAddressLine(), 392
 - getBestProvider(), 389
 - getBondedDevices(), 495
 - getCallState(), 428
 - getCenter(), 406
 - getConfiguredNetworks(), 493
 - getDefault(), 433
 - getDefaultSensor(), 488
 - getDesiredMinimumHeight(), 417
 - getDesiredMinimumWidth(), 417
 - getDisplayMessageBody(), 436
 - getDrawable(), 417
 - getDuration(), 421
 - getFeatureName(), 392
 - getFromLocationName(), 393
 - getHolder(), 445
 - getItem(), 341
 - getItemId(), 341
 - getLocality(), 392
 - getMaxAddressLineIndex(), 392
 - getMaxZoom(), 415
 - getOrientation(), 490
 - getResources(), 110
 - getScaleFactor(), 606
 - getSharedPreferences(), 111
 - getSystemService(), 428, 507
 - getText(), 194
 - getType(), 351
 - getView(), 342
 - getX(), 607
 - getY(), 607
 - getZoom(), 415
 - getZoomRatios(), 416
 - glColorPointer(), 453
 - glDrawElements(), 454, 455
 - glRotatef(), 451
 - gluLookAt(), 451
 - gluPerspective(), 450
 - GLUtils.texImage2D(), 459
 - glVertexPointer(), 452, 453
 - insert(), 309, 348
 - insertImage(), 416
 - isDiscovering(), 496
 - isEnabled(), 495
 - isFinishing(), 117
 - isRouteDisplayed(), 396
 - isSmoothZoomSupported(), 415
 - isZoomSupported(), 415
 - javaThrowsException(), 482
 - kaskadowa, 648
 - niebezpieczeństwa, 648, 649
 - listen(), 429, 431
 - Log.d(), 97

metoda

- Log.e(), 97
- Log.i(), 97
- Log.v(), 97
- Log.w(), 97
- managedQuery(), 332, 334, 561
- MediaPlayer.create(), 99
- MediaPlayer.release(), 99
- MediaPlayer.start(), 99
- MediaPlayer.stop(), 99
- najmniejszego wspólnego mianownika, 650
- obtainTypedArray(), 173
- onAccuracyChanged(), 488
- onActivityResult(), 593
- onAnimateMove(), 601
- onAnimateStep(), 601
- onBackup(), 578
- onClick(), 202
- onCreate(), 113, 114, 380, 436, 521, 546, 560
- onCreateDialog(), 226
- onCreateOptionsMenu(), 215
- onDeleted(), 538
- onDestroy(), 116, 117, 521, 546
- onDisabled(), 538
- onDraw(), 267, 599
- onEnabled(), 538
- onFling(), 602
- onKeyDown(), 463
- onKeyUp(), 463
- onLocationChanged(), 390
- onNewIntent(), 560
- onOffsetsChanged(), 547
- onPageFinished(), 375
- onPause(), 115
- onReceived(), 538
- onResetLocation(), 601, 602
- onRestore(), 578
- onResume(), 114, 115, 116
- onSaveInstanceState(), 116
- onScaleBegin(), 606
- onSensorChanged(), 488, 489
- onServiceConnected(), 528, 529
- onStart(), 521, 522, 523
- onStartCommand(), 521, 522, 523
- onStop(), 116
- onSurfaceChanged(), 546
- onSurfaceCreated(), 467, 469, 546
- onSurfaceDestroyed(), 546
- onTouchEvent(), 547, 598, 599, 601, 604
- onTrackballEvent(), 607
- onUpdate(), 538, 540
- onVisibilityChanged(), 546
- openFileOutput(), 302
- openOrCreateDatabase(), 307
- peekDrawable(), 417
- PictureCallback, 414
- post(), 460
- query(), 314, 558, 568
- recycle(), 459
- registerListener(), 488, 490
- removeDialog(), 225
- requestRestore(), 581
- requestStop(), 447
- requestRouteToHost(), 368
- run(), 447
- sendMultipartTextMessage(), 436
- sendTextMessage(), 433
- setBitmap(), 417
- setBuildInZoomControls(), 374
- setChronometer(), 539
- setContentView(), 189, 210
- setEGLContextClientVersion(), 469
- setFilters(), 198
- setImageURI(), 343
- setImageViewResource(), 539
- setInitialScale(), 374
- setJavaScriptEnabled(), 374
- setLatestEventInfo(), 515
- setLightTouchEnabled(), 375
- setOnClickListener(), 202
- setOnClickPendingIntent(), 540
- setOnCompletionListener(), 421
- setOnLongClickListener(), 223
- setPowerRequirement(), 389
- setProgressbar(), 539
- setResource(), 417
- setResult(), 566
- setShort(), 539
- setStream(), 417
- setString(), 539

- setSupportZoom(), 374
 - setText(), 190, 194
 - setTextViewText(), 539
 - setTheme(), 230
 - setVideoURI(), 421
 - setZoom(), 416
 - show(), 689
 - showDialog(), 225, 226
 - SmsManager.divideMessage(), 436
 - SmsMessage.createFromPdu(), 436
 - startActivity(), 118, 395, 438
 - startActivityForResult(), 495, 566, 593
 - startDiscovery(), 496
 - startService(), 121, 520, 526
 - stopSelf(), 520
 - surfaceChanged(), 412
 - surfaceCreated(), 446
 - surfaceDestroyed(), 412, 446
 - takePicture(), 414
 - unbindService(), 528
 - update(), 348
 - updateAppWidget(), 540
 - Uri.parse(), 99, 343
 - WebChromeClient.onJSAlert(), 382
 - wprowadzania tekstów, 585
 - MIME, 550, 551
 - minSdkVersion, 133
 - MMS, 41
 - MobiHand, 726
 - monitorowanie siły sygnału, 431
 - monitorowanie szybkości połączenia, 431
 - MotionEvent, 598, 599
 - Motorola
 - DynaTAC, 38
 - StarTAC, 39
 - MultiAutoCompleteTextView, 195, 196, 197
 - multimedia
 - dzwonki, 425
 - dźwięki, 409
 - obrazy, 409, 410
 - poszukiwanie, 425
 - rejestracja dźwięków, 421
 - wideo, 409, 418
 - Multimedia Messaging Service, *Patrz* MMS
 - multi-touch, 596
- ## N
- NDK, 442, 475
 - tworzenie projektu, 478
 - NinePatch, 275
 - Nine-Patch Stretchable Graphics, 165, 617, 618
 - Nine-Patch Stretchable Image, 164
 - NotePad, 74
 - NotificationManager, 507
 - numer wersji aplikacji, 712
- ## O
- obiekt
 - AdapterView, 258
 - ArrayAdapter, 256
 - CursorAdapter, 257
 - ListActivity, 259
 - TextView, 256
 - obroty, 291
 - obsługa
 - konfiguracji sprzętowych, 639
 - konfiguracji urządzeń, 638
 - numerów telefonicznych, 431
 - obsługa zmian konfiguracji, 631
 - ochrona własności intelektualnej, 726
 - odczyt
 - danych, 303
 - plików XML, 303
 - zawartości plików bajt po bajcie, 303
 - odczytywanie tekstów, 593
 - odkrywanie urządzeń, 496
 - odpowiedzi dotykowe, 589
 - ograniczenia urządzeń przenośnych, 664
 - okna dialogowe, 224, 225
 - AlertDialog, 224
 - CharacterPickerDialog, 224
 - DatePickerDialog, 224
 - definiowanie, 226
 - Dialog, 224
 - inicjalizacja, 226
 - ProgressDialog, 224
 - TimePickerDialog, 224
 - tworzenie, 227

- okna dialogowe
 - ukrywanie, 225, 227
 - uruchamianie, 226
 - usuwanie, 225, 227
 - wyświetlenie, 225
 - Określanie akcji intencji, 551
 - określanie rynków docelowych, 669
 - onDoubleTap, 598
 - onDoubleTapEvent, 598
 - onDown, 597
 - onFling, 598
 - onLongPress, 598
 - onScroll, 598
 - onServiceStateChanged(), 429
 - onShowPress, 597
 - onSingleTapConfirmed, 598
 - onSingleTapUp, 598
 - opcja
 - Debugowanie USB, 63
 - Programowanie, 63
 - OpenGL ES, 442, 443, 444, 468
 - zwalnianie zasobów, 464
 - 1.0, 442, 443
 - 1.1, 442, 443
 - 1.x, 467, 469
 - 2.0, 442, 443, 467, 469, 651
 - konfiguracja aplikacji, 468
 - pobieranie powierzchni, 468
 - opłata rejestracyjna, 719
 - oprogramowanie mobilne
 - metodologia, 648
 - projektowanie interfejsu, 648
 - tworzenie, 647
 - org.w3c.dom, 304
 - org.xml.sax.*, 304
 - org.xmlpull.*, 304
 - orientacja ekranu, 607
 - oświetlanie sceny, 455
 - OvalShape, 281
 - OvershootInterpolator, 293
- P**
- Paint, 267, 269
 - Parcel, 531
 - Parcelable, 529, 530
 - PathShape, 284
 - PendingIntent, 433
 - Perforce, 663
 - PetListAdapter, 343
 - PhoneNumberFormattingTextWatcher, 431
 - PhoneStateListener, 431
 - pielęgnacja oprogramowania mobilnego, 669
 - plik
 - arrays.xml, 172
 - dystrybucyjnego pakietu aplikacji, 720
 - manifestu, 125, 712
 - default.properties, 89
 - katalog src, 89
 - AndroidManifest.xml, 89
 - main.xml, 89
 - strings.xml, 89
 - R.java, 110, 146, 172
 - strings.xml, 172
 - układu, 235
 - edycja, 126
 - ręczna edycja, 129
 - zarządzanie tożsamością aplikacji, 131
 - XML, 170
 - zasobów, 146
 - pobieranie informacji o usłudze, 430
 - pobieranie opłat od użytkowników, 727
 - point-to-point, 494
 - pojemnik, 255
 - pojemniki działające w oparciu o dane, 255
 - polecenie SQL CREATE TABLE, 307
 - polityka zwracania aplikacji, 723
 - połączenie RFCOMM, 494
 - połączenie typu punkt-punkt, 494
 - Portable Network Graphics (PNG), 164
 - powiadomienia, 510
 - błyskanie, 512
 - domyślne działanie, 515
 - dźwięki, 514
 - projektowanie, 517
 - sygnalizator świetlny, 512
 - usuwanie, 511
 - wibracje, 511
 - poziom zabezpieczeń
 - niebezpieczny, 141
 - normalny, 141
 - poziom podpisu, 141

pozorowanie lokalizacji, 64
 preferencje

- aktualizacja, 299
- aktywności, 298
- dodawanie, 299
- odczyt, 299
- prywatne, 298
- przeszukiwanie, 299
- usuwanie, 299
- wspólne, 298

 PrefListenerService, 541
 problemy ze zgodnością, 613
 programowe definiowanie kształtów, 278
 programy certyfikacyjne, 662
 ProgressBar, 435
 progressBarStyleLarge, 209
 progressBarStyleSmall, 209
 projektowanie aplikacji

- bezpiecznych, 678
- błędy, 683
- interfejs użytkownika, 675
- korzystanie ze standardów, 680
- maksymalizacja zysków, 679
- najlepsze praktyki, 673, 683
- pod kątem aktualizacji, 680
- pod kątem łatwości aktualizacji
 - i rozszerzania, 681
- pod kątem prostoty ich utrzymania
 - przy wykorzystaniu Androida, 682
- stabilnych i szybko reagujących, 676

 przekształcanie

- kanału alfa, 291
- przesunięcia, 292
- przezroczystości, 291
- skali, 292
- tekstu na mowę, 589

 przesyłanie aplikacji do Android Market, 720
 przetwarzanie asynchroniczne, 362
 przewidywanie tekstu, 589
 przewijanie w pionie, 264
 przewijanie w poziomie, 264
 przytrzymanie, 222
 publikacja aplikacji w Android Market, 722
 punkt wstrzymania, 95

Q

queueEvent(), 467

R

RadialGradient, 270, 271
 RadioButton, 200, 201, 204
 RadioGroup, 201, 204
 RatingBar, 212
 rawQuery(), 316
 READ_PHONE_STATE, 428
 RecognizerIntent, 591, 592, 593
 RectShape, 279
 rejestracja

- dostawców treści, 139
- odbiorców komunikatów, 139
- usług, 139

 rejestracja głosu, 591
 rekordy

- aktualizacja, 309, 337
- dodawanie, 336
- usuwanie, 309, 310, 338
- wstawianie, 309

 RelativeLayout, 235, 539, 616
 RemoteViews, 537, 539
 Renderer, 443
 rendering, 444
 rendering pipeline, 444
 RoundRectShape, 279
 rozpoznawanie mowy, 589, 590, 591
 rysowanie

- bardziej złożonych obiektów, 453
- kwadratów, 279
- łuków, 282
- obiektów trójwymiarowych, 452
- owali, 281
- okręgów, 281
- prostokątów z zaokrąglonymi
 - wierzchołkami, 279
- prostokątów, 279
- różnych kształtów, 279
- ścieżek, 283
- wierzchołków, 452

S

- ScaleGestureDetector, 597, 604
- scenariusze testowania, 702
- ScrollView, 264, 616
- SearchManager, 555
- searchSuggestAuthority, 556
- SeekBar, 211
- Sensor, 487
- SensorManager, 486
- ServiceState, 429
- Settings, 328, 336
- ShapeDrawable, 281, 284
- SharedPreferences, 298, 299
- SharedPreferences.contains(), 299
- SharedPreferences.edit(), 299
- SharedPreferences.Editor.clear(), 300
- SharedPreferences.Editor.commit (), 300
- SharedPreferences.Editor.putBoolean(), 300
- SharedPreferences.Editor.putFloat(), 300
- SharedPreferences.Editor.putInt(), 300
- SharedPreferences.Editor.putLong(), 300
- SharedPreferences.Editor.putString(), 300
- SharedPreferences.Editor.remove(), 300
- SharedPreferences.getAll(), 299
- SharedPreferences.getBoolean(), 299
- SharedPreferences.getFloat(), 299
- SharedPreferences.getInt(), 299
- SharedPreferences.getLong(), 299
- SharedPreferences.getString(), 299
- SharedPreferencesBackupHelper, 578
- Short Messaging Service, Patrz SMS
- Short Text Message, 427
- showVoiceSearchButton, 558
- sigle-touch, 596
- simple_fragment, 469
- simple_vertex, 469
- SimpleCursorAdapter, 322, 561
- SimpleDataUpdateService, 541
- SimpleSearchableActivity, 560
- SimpleViewDetailsActivity, 552
- skalowanie, 292
- SlideME, 725
- SlidingDrawer, 265, 266
- słowniki użytkownika, 589
- SMS, 41, 427
- SmsManager, 433
- Snake, 74
 - dodawanie aplikacji, 78
 - pliki projektu, 80
 - uruchamianie aplikacji, 84
- SoftKeyboard, 588
- Software Development Kit (SDK), 46
- SoundBack, 590
- Spinner, 199
- splash screen, 111
- sprzedawanie aplikacji, 717, 725
- SQLite, 305, 308
- sqlite3, 689
- SQLiteDatabase, 307
- SQLiteOpenHelper, 319
- SQLiteQueryBuilder, 316
- standardowe kliknięcie, 222
- standardy kodowania, 685
 - definiowanie, 685
- START_TAG, 361
- stos aktywności, 112, 113
- stosowanie zasobów alternatywnych, 629
- style, 178, 227, 269
- Subversion, 663
- surfaceCreated(), 412
- SurfaceHolder, 412
- SurfaceView, 443, 445, 446
- SurfaceView.Callback, 446
- SweepGradient, 270, 271
- synchronizacja danych, 575
- synchronizacja danych użytkownika, 573
- synteza mowy, 590
- system
 - Binary Runtime for Wireless, 42
 - BREW, 42
 - Garnet OS, 42
 - GPS, 387
 - iPhone OS, 42
 - J2ME, 42
 - Java Micro Edition, 42
 - numeracji wersji aplikacji, 663
 - Palm OS, 42
 - rejestracji defektów, 691
 - RIM BlackBerry OS, 42

Symbian, 42
 zarządzania kodem źródłowym, 663
 CVS, 663
 Perforce, 663
 Subversion, 663

Ś

śledzenie informacji o awariach, 669
 środowisko programistyczne
 konfiguracja, 61
 przygotowanie, 61
 testowanie, 77
 zarządzanie, 693

T

TabActivity, 260
 TableLayout, 235, 616
 tablice indeksów, 453
 TalkBack, 590
 targetSdkVersion, 133, 134
 tekst
 określanie wymiarów, 275
 wyświetlanie, 272
 telefon
 Candy bar, 40
 klasyczny, 40
 przesuwany, 40
 z klapką, 40
 TelephonyManager, 428, 431
 tematy, 181, 229
 testy
 aktualizacji oprogramowania
 układowego, 670
 aplikacji klient-serwer, 661
 aplikacji na urządzenia przenośne, 668
 aplikacji na urządzeniach, 660
 aplikacji wielojęzycznych, 703
 aplikacji, 693
 automatyzacja, 697
 instalacji, 703
 jednostkowe, 687
 konfiguracja urządzenia, 63

kopii zapasowych, 703
 maksymalizacja pokrycia, 696
 na urządzeniach, 697
 najlepsze praktyki, 691
 narzędzia testowe innych firm, 662
 opłat za aplikację, 704
 pod kątem zwiększenia
 prawdopodobieństwa stworzenia
 aplikacji, która będzie hitem, 704
 pojedynczej aktywności, 687
 punktów integracji aplikacji, 702
 scenariusze, 702
 strategię, 699
 strukturalne, 699
 uaktualnień aplikacji, 702
 usług i serwerów używanych
 przez aplikacje mobilne, 700
 usług, 687
 użyteczności, 701
 w emulatorze, 697, 698, 699
 wizualnej atrakcyjności aplikacji
 wstępne, 697
 wydajności działania, 687, 704
 zdarzeń nieprzewidzianych, 704
 zgodności, 703
 TEXT, 361
 Text-To-Speech (TTS), 593
 TextView, 189, 227, 432
 themes, 181
 thumbnail, 182, 548
 TimePicker, 208
 Toast, 689
 ToggleButton, 200, 201
 Tokenizer, 195
 touch mode, 219
 trackball, 607
 trackListener, 524
 trackpad, 607
 transakcja, 311
 TreeViewObserver, 220
 TTS, 593, 594
 tworzenie aplikacji
 błędy, 689
 metodologia, 648

tworzenie

- baz danych SQLite, 307
- obiektu bazy danych, 307
- typy ekranów, 617
- typy wprowadzania danych, 587

U

układ, 188, 235

- AbsoluteLayout, 251
- FrameLayout, 243, 244, 245, 247, 248, 251
- RelativeLayout, 243, 247, 249, 250
- TableLayout, 243, 251, 252
- tworzenie, 235

ułatwienia dostępu, 589

- URI, 342
- dodawanie, 345

UriMatcher, 347

URLUtil, 378

UserDictionary, 328, 335

usługi, 110, 519, 520

- tworzenie, 521

ustawienia

- językowe, 634
- lokalne, 637

usuwanie aplikacji z Android Market, 723

usuwanie instancji bazy danych SQLite, 318

usuwanie rekordów z baz danych, 309

- usuwanie tabel oraz innych obiektów baz danych SQLite, 317

V

vertex shader, 467

Video.Media, 328

videoRecorder, 419

VideoView, 421

View, 234, 238

VIEW, 550

ViewGroup, 234, 235, 238, 239

- atrybuty, 244

viewport, 450

ViewTreeObserver, 221

voiceSearchMode, 558

W

WAP, 39, 40, 41

wartości logiczne, 298

wdrażanie aplikacji mobilnych, 668

WebBackForwardList, 378

WebChromeClient, 375, 376

WebHistoryItem, 378

WebView, 371, 372

weight, 616

weryfikacja informacji o awariach, 669

weryfikacja uprawnień aplikacji, 713

wibracje, 589

widoki, 187

widżet, 536, 537

- aktualizacja, 540

- hosty, 544

- instalacja, 543

- tworzenie dostawcy, 537

- tworzenie, 534, 535

wielkości ekranu, 137

wielodotyk, 596, 604

Wireless Application Protocol, 40, *Patrz* WAPWireless Markup Language, *Patrz* WML

WML, 40

writeToParcel(), 530

wstawianie rekordów, 309

wyjątki, 480

wykrywanie ruchów, 596

wymagania aplikacji, 659

wyszukiwanie

- globalne, 562, 563, 564

- głosowe, 558

- w pliku manifestu, 561

X

XML, 360

Z

zabezpieczenie przed kopiowaniem, 721

zakładka

- Application, 127, 128

- Instrumentation, 129

- Manifest, 127
 - Permissions, 127, 128
 - zamykanie bazy danych, 318
 - zamykanie i usuwanie baz danych SQLite, 317
 - zarządzanie kontami, 573
 - zarządzanie tożsamością aplikacji, 131
 - zasoby, 145, 146, 157
 - alternatywne, 150, 622
 - animacje poklatkowe, 148
 - animacje przejść, 148
 - animacje, 166
 - domyślne, 150
 - graficzne, 146, 163
 - grafika, 148
 - katalogi, 146
 - kolory, 147, 161
 - liczby całkowite, 147, 160
 - lista stanów koloru, 147
 - łańcuchy znaków w liczbie mnogiej, 147
 - łańcuchy znaków, 147, 156
 - menu, 148
 - odwołania do zasobów, 171
 - pliki dowolne, 148
 - pliki xml, 148
 - programowy dostęp, 152
 - proste zasoby graficzne, 148
 - style i tematy, 148
 - tablice liczb całkowitych, 148
 - tablice łańcuchów znaków, 147
 - tablice o wartościach różnych typów, 148
 - typy wartości, 147
 - układy, 148
 - wartości logiczne, 147, 159, 160
 - wymiary, 147
 - związane z interfejsem użytkownika, 146
 - zdalne tworzenie kopii, 576
 - zdarzenie
 - GlobalFocusChange, 221
 - GlobalLayout, 221
 - PreDraw, 221
 - zgodności aplikacji, 613
 - zmiana ustawień językowych, 634
 - znacznik
 - <action>, 553
 - <activity>, 553, 561, 569
 - <alpha>, 148
 - <animation-list>, 148
 - <application>, 142
 - <array>, 148
 - <bool>, 147
 - <category>, 553
 - <color>, 147
 - <dimen>, 147
 - <drawable>, 148
 - <instrumentation>, 142
 - <integer>, 147
 - <integer-array>,, 148
 - <intent-filter>, 142, 553
 - <item>, 147, 148
 - <menu>, 148
 - <meta-data>, 561
 - <plurals>, 147, 159
 - <receiver>, 142, 542
 - <resources>, 149
 - <rotate>, 148
 - <scale>, 148
 - <selector>, 147
 - <service>, 520
 - <set>, 148
 - <string>, 147
 - <string-array>, 147
 - <style>, 148, 178, 181
 - <supports-screen>, 639
 - <TextView>, 189
 - <translate>, 148
 - <uses-configuration>, 135, 136, 607, 639
 - <uses-feature>, 136, 443, 468, 483, 639
 - <uses-sdk>, 133, 134, 468, 483, 548, 639
- Ż**
- żądanie
 - odtworzenia danych, 581
 - utworzenia kopii, 581



**WYCZERPUJĄCY PRZEWODNIK,
GRUNTOWNIE ZAKTUALIZOWANY
W OPARCIU O NAJNOWSZĄ WERSJĘ ANDROID SDK
I NAJLEPSZE TECHNIKI PROGRAMOWANIA!**

Android

PROGRAMOWANIE APLIKACJI NA URZĄDZENIA PRZEŃOŚNE

Gdy w 2008 roku Google opublikował Androida, rynkiem telefonów komórkowych wprowadził niasco zatrzędo, ale nikt nie spodziewał się, że ta platforma aż tak poważnie zagrozi takim gigantom mobilnych systemów operacyjnych, jak iOS Apple, Windows Mobile, Symbian czy RIM BlackBerry. Od tego czasu każde nowe urządzenie z Androidem miało coraz większe możliwości i było jeszcze bardziej ekscytujące od swoich poprzedników. Minęły zaledwie trzy lata od dnia, kiedy na rynku pojawił się pierwszy telefon z systemem Android – T-Mobile G1, stworzony przez firmę HTC – a system ten już okrzyknięty został najszybciej sprzedającą się platformą dla telefonów przenośnych. To oczywiście nie pozostaje bez echa: potrzeba coraz więcej ludzi specjalizujących się w programowaniu aplikacji na tę fascynującą, darmową i otwartą platformę mobilną.

Oto kompletny podręcznik, zawierający wszystko, co potrzebne do tworzenia, wdrażania i sprzedawania aplikacji na urządzenia przenośne działające pod kontrolą najnowszych wersji Androida. Autorzy – w oparciu o swoje wieloletnie doświadczenie w tworzeniu mobilnych aplikacji – wprowadzą Cię we wszystkie etapy tego procesu: pomysł, projektowanie, pisanie kodu, testowanie, pakowanie i rozpowszechnianie aplikacji. Poznasz doskonale specyfikację platformy Android, podstawowe zasady efektywnego projektowania aplikacji na nią przeznaczonych oraz najlepsze praktyki związane z tworzeniem wygodnych interfejsów użytkownika. Znajdziesz tu także wyczerpujące opisy wszystkich kluczowych interfejsów programistycznych: do obsługi składowania danych, komunikacji sieciowej, obsługi rozmów telefonicznych, usług lokalizacyjnych, multimediów, grafiki 3D oraz opcjonalnych komponentów sprzętowych. Oprócz tego książka została uzupełniona praktycznymi sztuczkami, które pozwolą Ci zaoszczędzić sporo cennego czasu i uniknąć wielu niepotrzebnych pułapek!

Ponadto znajdziesz tu:

- kilka rozdziałów opisujących interfejs API do obsługi technologii związanych z WWW, a także Android NDK, poszerzanie zasięgu aplikacji, zarządzanie użytkownikami, synchronizację danych, tworzenie kopii bezpieczeństwa, zaawansowane metody wprowadzania danych
- wyczerpujące informacje na temat plików manifestu, dostawców treści, projektowania i testowania aplikacji
- prezentację najbardziej aktualnych i interesujących zagadnień, takich jak obsługa komunikacji Bluetooth i gestów, rozpoznawanie mowy, widżety, technologie Live Folders, Live Wallpapers oraz globalne wyszukiwanie
- aktualne informacje na temat generowania grafiki 3D przy użyciu OpenGL ES 2.0
- zagadnienia związane z zapewnianiem zgodności pomiędzy różnymi urządzeniami

Shane Conder ma szerokie doświadczenia programistyczne. Zaprojektował i napisał wiele komercyjnych aplikacji na Androida, iPhone'a, BREW, BlackBerry, Palm oraz środowiska J2ME i Windows Mobile; część z nich została zainstalowana na milionach telefonów działających na całym świecie.

Lauren Darcy jest kierowniczką do spraw technicznych oraz wyznacza kierunki działalności w firmie programistycznej zajmującej się technologiami mobilnymi (w tym Androidem, iPhone'em, BlackBerry czy J2ME) i działalnością konsultingową. Dysponując ponaddwudziestoletnim doświadczeniem, Lauren uznawana jest za niekwestionowany autorytet w dziedzinie architektury aplikacji oraz tworzenia komercyjnych aplikacji mobilnych.

helion.pl
księgarnia
internetowa

Nr katalogowy: 6616



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900
0 601 339900



Helion

Sprawdź najnowsze promocje:

➔ <http://helion.pl/promocje>

➔ Książki najchętniej czytane:

➔ <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

➔ <http://helion.pl/nowosci>

Helion SA

ul. Kosciuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-246-3349-4



Cena 149,00 zł

Informatyka w najlepszym wydaniu

9 788324 633494