



Kyle Richter  
Joe Keeley

# iOS Tajniki SDK

Biblioteka przydatnych narzędzi

Poznaj zaawansowane możliwości platformy iOS!



Tytuł oryginału: iOS Components and Frameworks: Understanding the Advanced Features of the iOS SDK

Tłumaczenie: Krzysztof Rychlicki-Kicior

ISBN: 978-83-246-9182-1

Authorized translation from the English language edition, entitled:  
IOS COMPONENTS AND FRAMEWORKS: UNDERSTANDING  
THE ADVANCED FEATURES OF THE IOS SDK; ISBN 0321856716;  
by Kyle Richter; and by Joe Keeley; published by Pearson Education, Inc,  
publishing as Addison Wesley.  
Copyright © 2014 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.  
Polish language edition published by HELION S.A. Copyright © 2014.

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

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

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

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

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

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

Printed in Poland.

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

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

# Spis treści

<b>Słowo wstępne</b> .....	<b>17</b>
<b>Przedmowa</b> .....	<b>19</b>
<b>O autorach</b> .....	<b>27</b>
<b>Rozdział 1. UIKit Dynamics</b> .....	<b>29</b>
Przykładowa aplikacja .....	29
Wprowadzenie do UIKit Dynamics .....	30
Implementacja mechanizmów UIKit Dynamics .....	31
Grawitacja .....	32
Kolizje .....	33
Powiązania .....	35
Sprężynowanie .....	36
Przyciąganie .....	37
Siły odpychania .....	37
Właściwości elementu .....	39
UIDynamicAnimator i UIDynamicAnimatorDelegate pod lupą .....	41
Podsumowanie .....	41
Ćwiczenia .....	42
<b>Rozdział 2. Core Location, MapKit i wirtualne granice</b> .....	<b>43</b>
Przykładowa aplikacja .....	43
Uzyskiwanie położenia użytkownika .....	44
Wymagania i uprawnienia .....	44
Sprawdzanie dostępności usług .....	48
Rozpoczęcie pobierania informacji o położeniu .....	48
Parsowanie i interpretacja danych położenia .....	50
Powiadomienia o istotnych zmianach .....	52
Stosowanie plików GPX do testowania lokalizacji .....	52
Wyświetlanie map .....	53
Układy współrzędnych .....	53
Konfiguracja MKMapView .....	54
Reagowanie na zachowania użytkownika .....	56

Adnotacje i warstwy .....	56
Dodawanie adnotacji .....	57
Wyświetlanie standardowych i specyficznych widoków adnotacji ...	59
Przeciągane widoki adnotacji .....	62
Praca z warstwami mapy .....	63
Geokodowanie i odwrotne geokodowanie .....	65
Geokodowanie adresu .....	65
Odwrotne geokodowanie położenia .....	68
Wirtualne granice .....	71
Weryfikacja możliwości w zakresie monitorowania obszaru .....	71
Określanie granic .....	72
Monitorowanie zmian .....	72
Wyznaczanie tras .....	74
Podsumowanie .....	79
Ćwiczenia .....	79
<b>Rozdział 3. Tablice wyników .....</b>	<b>81</b>
Uderz kaktus .....	81
Wyświetlanie kaktusa .....	83
Interakcja z kaktusami .....	85
Wyświetlanie liczby żyć i wyniku .....	87
Wstrzymywanie i wznowianie .....	89
Uwagi końcowe .....	90
iTunes Connect .....	90
Menedżer Game Center .....	94
Uwierzytelnianie .....	95
Typowe błędy związane z uwierzytelnianiem .....	96
Uwierzytelnianie w iOS 6 .....	97
Wysyłanie wyników .....	99
Dodawanie wyników do gry Whack-a-Cac .....	102
Wyświetlanie list wyników .....	104
Wyzwania .....	106
Co dalej z tablicami wyników? .....	108
Podsumowanie .....	109
Ćwiczenia .....	109
<b>Rozdział 4. Osiągnięcia .....</b>	<b>111</b>
iTunes Connect .....	111
Wyświetlanie informacji o osiągnięciach .....	114
Menedżer Game Center i uwierzytelnianie .....	115
Pamięć podręczna osiągnięć .....	115
Zgłaszanie osiągnięć .....	117
Dodawanie haków osiągnięć .....	119
Banery z powiadomieniami .....	120

Wyzwania .....	120
Dodawanie osiągnięć do gry .....	123
Ukończone i nieukończone osiągnięcia .....	124
Osiągnięcia częściowe .....	125
Osiągnięcia zdobywane na przestrzeni wielu sesji .....	127
Grupowanie osiągnięć i przechowywanie ich dokładności .....	127
Osiągnięcia z wykorzystaniem zegarów .....	128
Resetowanie osiągnięć .....	129
Co jeszcze można osiągnąć? .....	130
Podsumowanie .....	132
Ćwiczenia .....	132
<b>Rozdział 5. Książka adresowa .....</b>	<b>133</b>
Czemu obsługa książki adresowej jest istotna? .....	133
Ograniczenia związane ze stosowaniem książki adresowej .....	134
Wprowadzenie do przykładowej aplikacji .....	134
Dodanie obsługi książki adresowej .....	135
Wczytywanie danych z książki adresowej .....	137
Odczytywanie wielowartości z książki adresowej .....	139
Etykiety w książce adresowej .....	139
Obsługa adresów .....	140
Interfejs graficzny książki adresowej .....	141
Kontrolka wyboru kontaktów .....	142
Tworzenie kontaktów z poziomu kodu .....	147
Podsumowanie .....	148
Ćwiczenia .....	149
<b>Rozdział 6. Obsługa bibliotek muzyki .....</b>	<b>151</b>
Wprowadzenie do przykładowej aplikacji .....	151
Tworzenie silnika odtwarzania .....	152
Rejestrowanie powiadomień odtwarzania .....	153
Kontrolki użytkownika .....	154
Obsługa zmiany stanu .....	156
Czas trwania a zegary .....	160
Tryb losowy i powtarzanie .....	161
Media Picker — kontrolka wyboru multimediiów .....	162
Ręczny wybór piosenek .....	164
Odtwarzanie losowo wybranej piosenki .....	164
Wyszukiwanie piosenek w oparciu o predykaty .....	166
Podsumowanie .....	167
Ćwiczenia .....	168
<b>Rozdział 7. Obsługa i parsowanie formatu JSON .....</b>	<b>169</b>
JSON .....	169
Korzyści wynikające ze stosowania formatu JSON .....	170
Zasoby JSON .....	170

Omówienie przykładowej aplikacji .....	170
Uzyskanie dostępu do serwera .....	171
Pobieranie danych w formacie JSON z serwera .....	171
Tworzenie obiektu żądania .....	171
Przetwarzanie odpowiedzi .....	172
Parsowanie danych w formacie JSON .....	173
Wyświetlanie danych .....	174
Wysyłanie ogłoszenia .....	175
Przekształcanie danych na format JSON .....	176
Wysyłanie dokumentu JSON do serwera .....	178
Podsumowanie .....	179
Ćwiczenie .....	179
<b>Rozdział 8. iCloud w praktyce .....</b>	<b>181</b>
Przykładowa aplikacja .....	182
Konfiguracja aplikacji dla usługi iCloud .....	182
Konfiguracja konta .....	182
Włączenie obsługi iCloud .....	184
Inicjalizacja usługi iCloud .....	184
Wprowadzamy klasę UIDocument .....	185
Tworzenie podklas klasy UIDocument .....	186
Interakcja z obiektem klasy UIDocument .....	187
Interakcja z usługą iCloud .....	188
Wyświetlanie dokumentów w usłudze iCloud .....	188
Wykrywanie konfliktów w usłudze iCloud .....	191
Rozwiązywanie konfliktów .....	193
Synchronizacja słowników .....	197
Podsumowanie .....	199
Ćwiczenia .....	199
<b>Rozdział 9. Powiadomienia .....</b>	<b>201</b>
Różnice pomiędzy powiadomieniami lokalnymi i zdalnymi .....	201
Przykładowa aplikacja .....	202
Konfiguracja aplikacji .....	203
Tworzenie programistycznego certyfikatu SSL dla zdalnych powiadomień .....	206
Programistyczny profil poświadczeń .....	209
Przygotowanie własnego dźwięku .....	213
Rejestrowanie aplikacji w celu otrzymywania zdalnych powiadomień .....	214
Planowanie powiadomień lokalnych .....	215
Otrzymywanie powiadomień .....	216
Serwer powiadomień zdalnych .....	217
Podstawowa konfiguracja Rails .....	218
Dodawanie obsługi urządzeń i wykrzykników .....	219

Kontroler urządzeń .....	222
Kontroler wykrzyknień .....	222
Połączenie aplikacji mobilnej i serwera .....	223
Wysyłanie zdalnych powiadomień .....	226
Obsługa informacji zwrotnej od APNs .....	227
Podsumowanie .....	227
Ćwiczenie .....	227
<b>Rozdział 10. Bluetooth a Game Kit .....</b>	<b>229</b>
Ograniczenia komunikacji sieciowej Bluetooth w Game Kit .....	229
Korzyści ze stosowania Bluetooth w Game Kit .....	230
Przykładowa aplikacja .....	230
Wybór partnera — kontrolka Peer Picker .....	234
Wysyłanie danych .....	238
Tryby transferu danych .....	238
Wysyłanie danych w przykładowej aplikacji .....	239
Otrzymywanie danych .....	240
Otrzymywanie danych w przykładowej aplikacji .....	240
Zmiany stanu .....	241
Zaawansowane funkcje .....	242
Wyświetlanie nazwy partnera .....	242
Nawiązywanie połączenia bez kontrolki Peer Picker .....	242
Tryby sesji .....	244
Podsumowanie .....	244
Ćwiczenia .....	244
<b>Rozdział 11. AirPrint .....</b>	<b>245</b>
Drukarki AirPrint .....	245
Testowanie technologii AirPrint .....	247
Drukowanie tekstu .....	247
Informacje na temat drukowania .....	248
Ustawienie zakresu stron .....	249
Obsługa błędów .....	250
Rozpoczynamy drukowanie .....	250
Informacja zwrotna od symulatora drukowania .....	251
Centrum drukowania (Print Center) .....	251
UIPrintInteractionControllerDelegate .....	253
Drukowanie wyrenderowanego dokumentu HTML .....	254
Drukowanie plików PDF .....	255
Podsumowanie .....	256
Ćwiczenia .....	256

<b>Rozdział 12. Wprowadzenie do Core Data .....</b>	<b>257</b>
Kiedy stosować Core Data? .....	258
Obiekty zarządzane Core Data .....	259
Obiekty zarządzane .....	259
Model zarządzanego obiektu .....	260
Migracje modeli zarządzanych obiektów .....	262
Tworzenie obiektów zarządzanych .....	263
Pobieranie i sortowanie obiektów .....	263
Kontroler pobranych wyników .....	265
Środowisko Core Data .....	265
Koordynator trwałego magazynu .....	265
Trwały magazyn .....	266
Kontekst obiektów zarządzanych .....	266
Podsumowanie .....	267
<b>Rozdział 13. Core Data w praktyce .....</b>	<b>269</b>
Przykładowa aplikacja .....	269
Tworzenie projektu wykorzystującego Core Data .....	270
Środowisko Core Data .....	272
Tworzenie modelu obiektu zarządzanego .....	274
Tworzenie encji .....	275
Dodawanie atrybutów .....	275
Definiowanie związków .....	276
Własne podklasy obiektów zarządzanych .....	277
Konfiguracja domyślnych danych .....	278
Wstawianie nowych obiektów zarządzanych .....	278
Inne sposoby konfiguracji domyślnych danych .....	279
Wyświetlanie obiektów zarządzanych .....	280
Tworzenie własnych żądań pobrania .....	280
Pobieranie za pomocą identyfikatora obiektu .....	282
Wyświetlanie danych obiektu .....	283
Stosowanie predykatów .....	284
Kontroler pobranych wyników .....	285
Przygotowanie kontrolera pobranych wyników .....	287
Integracja widoku tabeli z kontrolerem pobranych wyników .....	288
Reagowanie na zmiany w bibliotece Core Data .....	290
Dodawanie, edytowanie i usuwanie obiektów zarządzanych .....	293
Wstawianie nowego obiektu zarządzanego .....	293
Usuwanie obiektu zarządzanego .....	294
Edycja istniejącego obiektu zarządzanego .....	294
Zapisywanie i wycofywanie zmian .....	295
Podsumowanie .....	296
Ćwiczenia .....	297



<b>Rozdział 14. Mechanizmy języka .....</b>	<b>299</b>
Literały .....	300
NSNumber .....	300
NSArray .....	301
NSDictionary .....	301
Opakowania .....	302
Automatyczne zliczanie referencji .....	302
Stosowanie ARC w nowym projekcie .....	303
Konwersja istniejącego projektu na ARC .....	303
Jak stosować ARC .....	305
Kwalifikatory ARC .....	307
Bloki .....	307
Deklarowanie i stosowanie bloków .....	308
Przechwytywanie stanu za pomocą bloków .....	309
Bloki jako parametry metod .....	310
Pamięć, wątki i bloki .....	311
Właściwości .....	313
Deklarowanie właściwości .....	313
Syntetyzacja właściwości .....	315
Dostęp do właściwości .....	315
Notacja kropkowa .....	315
Szybkie wyliczenia .....	316
Podmiana metod .....	317
Podsumowanie .....	319
Ćwiczenia .....	320
<b>Rozdział 15. Integracja Twittera i Facebooka</b>	
<b>z wykorzystaniem frameworku społecznościowego .....</b>	<b>321</b>
Integracja z usługami społecznościowymi .....	321
Przykładowa aplikacja .....	322
Logowanie .....	323
Stosowanie klasy SLComposeViewController .....	324
Wysyłanie wiadomości za pomocą własnego interfejsu .....	326
Wysyłanie komunikatów do serwisu Twitter .....	327
Wysyłanie wiadomości do Facebooka .....	330
Tworzenie aplikacji do serwisu Facebook .....	330
Dostęp do osi czasu użytkownika .....	336
Twitter .....	337
Facebook .....	341
Podsumowanie .....	345
Ćwiczenia .....	345

<b>Rozdział 16. Obsługa zadań działających w tle .....</b>	<b>347</b>
Przykładowa aplikacja .....	348
Sprawdzanie dostępności zadań w tle .....	348
Kończenie zadania działającego w tle .....	349
Identyfikator zadania w tle .....	350
Metoda obsługi wygaśnięcia .....	350
Kończenie zadania w tle .....	351
Implementacja aktywności wykonywanych w tle .....	352
Rodzaje aktywności wykonywanych w tle .....	353
Odtwarzanie muzyki w tle .....	354
Podsumowanie .....	356
Ćwiczenia .....	357
<b>Rozdział 17. Grand Central Dispatch a wydajność .....</b>	<b>359</b>
Przykładowa aplikacja .....	359
Wprowadzenie do kolejek .....	361
Wykonywanie operacji w głównym wątku .....	361
Działanie w tle .....	363
Uruchamianie kolejki operacji .....	365
Operacje współbieżne .....	366
Operacje szeregowe .....	367
Anulowanie operacji .....	369
Własne operacje .....	369
Uruchamianie kolejki dyspozytora .....	371
Współbieżne kolejki dyspozytora .....	372
Szeregowe kolejki dyspozytora .....	373
Podsumowanie .....	375
Ćwiczenia .....	376
<b>Rozdział 18. Stosowanie pęków kluczy do zabezpieczania danych .....</b>	<b>377</b>
Wprowadzenie do przykładowej aplikacji .....	378
Konfiguracja i stosowanie pęku kluczy .....	378
Konfiguracja obiektu klasy KeychainItemWrapper .....	379
Przechowywanie i pobieranie PIN-u .....	380
Klucze atrybutów pęku kluczy .....	381
Zabezpieczanie słownika .....	382
Resetowanie elementu pęku kluczy .....	384
Współdzielenie pęku kluczy pomiędzy aplikacjami .....	384
Kody błędów pęków kluczy .....	385
Podsumowanie .....	386
Ćwiczenia .....	386
<b>Rozdział 19. Operacje na filtrach i obrazkach .....</b>	<b>387</b>
Przykładowa aplikacja .....	387
Zasady obsługi i wyświetlania obrazków .....	388

Tworzenie obiektu obrazka .....	388
Wyświetlanie obrazka .....	389
Stosowanie kontrolki Image Picker .....	391
Zmiana rozmiaru obrazka .....	394
Filtry biblioteki Core Image .....	395
Kategorie filtrów a filtry .....	395
Atrybuty filtra .....	397
Inicjalizacja obrazka .....	400
Renderowanie przefiltrowanego obrazka .....	401
Łączenie filtrów .....	401
Wykrywanie twarzy .....	403
Konfiguracja detektora twarzy .....	403
Przetwarzanie rysów twarzy .....	404
Podsumowanie .....	405
Ćwiczenia .....	406
<b>Rozdział 20. Widoki kolekcji .....</b>	<b>407</b>
Przykładowa aplikacja .....	407
Wprowadzamy widoki kolekcji .....	408
Konfigurowanie widoku kolekcji .....	408
Implementacja metod źródła danych dla widoku kolekcji .....	411
Implementacja metod delegacji widoku kolekcji .....	413
Dostosowywanie widoku kolekcji i układu przepływu .....	415
Podstawowe modyfikacje .....	415
Widoki dekoracyjne .....	416
Tworzenie własnych układów .....	420
Animacje widoku kolekcji .....	424
Zmiany układu widoku kolekcji .....	424
Animacje układu widoku kolekcji .....	426
Animacje zmiany widoku kolekcji .....	427
Podsumowanie .....	428
Ćwiczenia .....	428
<b>Rozdział 21. Wprowadzenie do UITextView .....</b>	<b>429</b>
Przykładowa aplikacja .....	430
Klasa UITextView .....	430
UITextView .....	431
Dynamiczne wykrywanie hiperłączy .....	433
Wykrywanie trafień .....	434
Ścieżki wyłączające .....	435
Podświetlanie uzależnione od treści .....	436
Zmiana ustawień czcionki dla typów dynamicznych .....	441
Podsumowanie .....	442
Ćwiczenia .....	442

<b>Rozdział 22. Rozpoznawanie gestów .....</b>	<b>443</b>
Rodzaje detektorów gestów .....	443
Sposoby użycia detektorów gestów .....	444
Wprowadzenie do przykładowej aplikacji .....	444
Detektor stuknięć .....	445
Detektor szczygnięć .....	446
Wiele detektorów w jednym widoku .....	448
Detektory gestów — z czym to się je .....	450
Wiele detektorów w jednym widoku — reduks .....	451
Wykrywanie nieudanych gestów .....	453
Tworzenie własnych podklas UIGestureRecognizer .....	454
Podsumowanie .....	455
Ćwiczenie .....	455
<b>Rozdział 23. Obsługa bibliotek zdjęć .....</b>	<b>457</b>
Przykładowa aplikacja .....	457
Biblioteka zasobów .....	458
Wyliczanie zasobów i grup zasobów .....	458
Uprawnienia .....	459
Grupy .....	460
Zbiory .....	464
Wyświetlanie zasobów .....	466
Zapisywanie na rolce aparatu .....	470
Obsługa strumienia zdjęć .....	472
Podsumowanie .....	474
Ćwiczenia .....	474
<b>Rozdział 24. Passbook i PassKit .....</b>	<b>475</b>
Przykładowa aplikacja .....	476
Projektowanie kuponu .....	477
Rodzaje kuponów .....	477
Bilet podróżny — układ .....	478
Kupon zniżkowy — układ .....	478
Wydarzenie — układ .....	478
Kupon ogólny — układ .....	479
Karta podarunkowa — układ .....	480
Wyświetlanie kuponu .....	481
Tworzenie kuponu .....	482
Identyfikacja kuponu .....	484
Istotne informacje na temat kuponu .....	484
Identyfikacja kodu kreskowego .....	485
Informacje na temat wyglądu kuponu .....	486
Pola kuponu .....	486

Podpisywanie i tworzenie paczki kuponu .....	489
Tworzenie identyfikatora typu kuponu .....	489
Tworzenie certyfikatu podpisywania kuponów .....	491
Tworzenie manifestu .....	495
Podpisywanie i tworzenie paczki dla kuponu .....	495
Testowanie kuponu .....	496
Interakcja z kuponami z poziomu aplikacji .....	497
Automatyczna aktualizacja kuponów .....	506
Podsumowanie .....	507
Ćwiczenia .....	507
<b>Rozdział 25. Debugowanie i narzędzia .....</b>	<b>509</b>
Wprowadzenie do debugowania .....	509
Pierwszy błąd w programie komputerowym .....	510
Podstawy debugowania w Xcode .....	510
Punkty przerwania .....	512
Dostosowywanie punktów przerwania .....	513
Punkty przerwania wyjątków i symboliczne punkty przerwania .....	514
Zasięg punktu przerwania .....	514
Praca z debuggerem .....	515
Dodatkowe narzędzia — instrumenty .....	517
Interfejs instrumentów .....	518
Odkrywamy instrumenty — profiler czasowy .....	520
Odkrywamy instrumenty — wycieki pamięci .....	522
Co dalej z instrumentami? .....	525
Podsumowanie .....	526
Ćwiczenia .....	526
<b>Skorowidz .....</b>	<b>527</b>



# Obsługa bibliotek muzyki

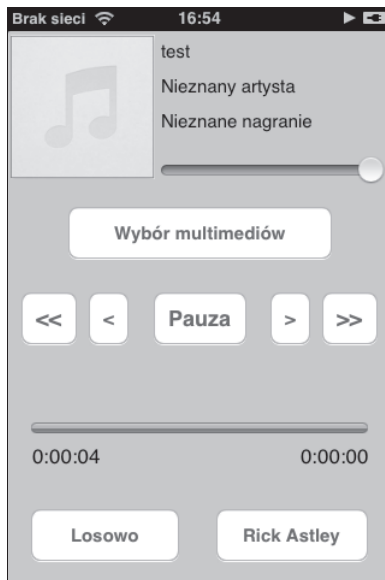
*Gdy Steve Jobs po raz pierwszy pokazał światu iPhone'a na konwencji Macworld w 2007 roku, został on ochrzczone mianem połączenia telefonu, iPod'a i rewolucyjnego komunikatora internetowego. Wiele lat później, po części dzięki ciężkiej pracy wielu programistów na całym świecie, iPhone stał się czymś więcej. Oryginalne przesłanie nie uległo jednak zmianie — iPhone cały czas pełni przede wszystkim funkcję telefonu, odtwarzacza muzyki i urządzenia do komunikacji w internecie. Użytkownicy nie dodali iPhone'a do kolekcji swoich urządzeń, które noszą każdego dnia — zastąpili swoje stare telefony i iPody jednym urządzeniem.*

*Odtwarzanie muzyki stanowiło istotny aspekt iPhone'a od powstania pierwszych koncepcji tego urządzenia, czyli od 2004 roku. iPhone był zawsze traktowany jako rozszerzenie iPod'a. Muzyka stała się również inspiracją do stworzenia iPod'a, który okazał się jednym z największych sukcesów firmy Apple. Ludzie kochają muzykę, bo łączy ludzi i pozwala na pełne wyrażanie siebie. Choć wielu użytkowników iPhone'ów może nie traktować swoich urządzeń jako odtwarzaczy, większość z nich bez większego zastanowienia słucha na nich swoich ulubionych piosenek.*

*W tym rozdziale nauczymy się korzystać z biblioteki muzyki użytkownika wewnątrz aplikacji iOS. Niezależnie od tego, czy chcesz stworzyć w pełni funkcjonalny odtwarzacz muzyki, czy też chcesz odtwarzać wybraną przez użytkownika muzykę w formie ścieżki dźwiękowej w grze, w tym rozdziale dowiesz się, jak to zrobić.*

## Wprowadzenie do przykładowej aplikacji

Przykładowa aplikacja w tym rozdziale nosi nazwę *Odtwarzacz* (rysunek 6.1). Jest to w pełni funkcjonalny odtwarzacz muzyki na iPhone'a. Użytkownik może wybierać utwory do odtworzenia przy użyciu komponentu Media Picker, może także odtwarzać utwory losowo lub wybrać konkretnych artystów. Co więcej, użytkownik będzie w stanie wstrzymywać utwór i powracać do niego, przechodzić pomiędzy nimi oraz regulować głośność odtwarzania. Ponadto zostanie zaimplementowany licznik, a dwa dodatkowe przyciski pozwolą na trzydziestosekundowe przeskoki w odtwarzanym utworze.



Rysunek 6.1. Pierwszy rzut oka na przykładową aplikację — w pełni funkcjonalny odtwarzacz muzyki działający na iPodzie

Co więcej, aplikacja wyświetli okładkę dla aktualnie odtwarzanej ścieżki, jeśli jest ona dostępna.

Emulator (symulator) systemu iOS dostarczony razem z Xcode nie zawiera aplikacji *Muzyka*. Nie możemy też przenieść plików muzycznych do jego systemu plików. Aplikacja zadziała tylko na urządzeniach. W momencie uruchomienia aplikacji w emulatorze zostanie wyświetlonych kilka błędów.

```
2013-03-02 16:04:13.392 player[80633:c07] MPMusicPlayer: Unable to launch iPod music
↳player server: application not found
2013-03-02 16:04:13.893 player[80633:c07] MPMusicPlayer: Unable to launch iPod music
↳player server: application not found
2013-03-02 16:04:14.395 player[80633:c07] MPMusicPlayer: Unable to launch iPod music
↳player server: application not found
```

Dowolna próba dostępu do biblioteki mediów spowoduje zawieszenie się symulatora z następującym komunikatem:

```
*** Terminating app due to uncaught exception 'NSInternalInconsistencyException',
↳reason: 'Unable to load iPodUI.framework'
```

## Tworzenie silnika odtwarzania

Zanim zaczniemy korzystać z jakichkolwiek danych audio, musimy zrozumieć, jakie kontrolki odtwarzania będą nam potrzebne. Odtworzenie muzyki z poziomu aplikacji



wymaga utworzenia nowej instancji klasy `MPMusicPlayerController`. Operację tę wykonujemy w pliku nagłówkowym `ICFViewController.h` — nowy obiekt nosi nazwę `player`. Klasa `MPMusicPlayerController` jest wykorzystywana w tym rozdziale do kontroli odtwarzania oraz do pobierania informacji o odtwarzanych elementach.

```
@interface ICFViewController : UIViewController
{
    MPMusicPlayerController *player;
}
```

```
@property (nonatomic, retain) MPMusicPlayerController *player;
```

Wewnątrz metody `viewDidLoad` odtwarzacz typu `MPMusicPlayerController` można zainicjalizować, korzystając z jednej z metod klasy `MPMusicPlayerController`. Pierwszy wariant — `applicationMusicPlayer` — odtworzy muzykę wewnątrz aplikacji. Nie wpłynie on na stan iPod'a, a odtwarzanie zakończy się w momencie zamknięcia aplikacji. Drugi wariant — metoda `iPodMusicPlayer` — pozwoli na kontrolę samego iPod'a. Odtwarzacz powróci do miejsca, w którym użytkownik pozostawił odtwarzanie na iPodzie, i skorzysta z tej samej ścieżki. Odtwarzanie będzie kontynuowane nawet po przejściu aplikacji w tryb pracy w tle. Przykładowa aplikacja wykorzystuje metodę `applicationMusicPlayer`, ale można zmienić to wywołanie, nie wprowadzając dodatkowego kodu ani zachowania.

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    player = [MPMusicPlayerController applicationMusicPlayer];
}
```

## Rejestrowanie powiadomień odtwarzania

Wydajne odtwarzanie muzyki wymaga znajomości stanów odtwarzacza. W związku z tym istnieje konieczność obserwacji trzech powiadomień. Są to powiadomienia o zmianie aktualnie odtwarzanego utworu, zmianie głośności i zmianie stanu odtwarzania. Stany te mogą być monitorowane dzięki zapisaniu się na wspomniane zdarzenia przy użyciu klasy `NSNotificationCenter`. Przykładowa aplikacja korzysta z użytecznej metody `registerMediaPlayerNotifications`, dzięki czemu kod aplikacji pozostaje prosty i czytelny. Po dodaniu nowych obserwatorów do klasy `NSNotificationCenter` należy wywołać metodę odtwarzacza `beginGeneratingPlaybackNotifications`.

```
- (void)registerMediaPlayerNotifications
{
    NSNotificationCenter *notificationCenter = [NSNotificationCenter defaultCenter];
    [notificationCenter addObserver: self
        selector: @selector(nowPlayingItemChanged:)
        name: MPMusicPlayerControllerNowPlayingItemDidChangeNotification
        object: player];
}
```

```

[notificationCenter addObserver: self
 selector: @selector (playbackStateChanged:)
 name: MPMusicPlayerControllerPlaybackStateDidChangeNotification
 object: player];

[notificationCenter addObserver: self
 selector: @selector (volumeChanged:)
 name: MPMusicPlayerControllerVolumeDidChangeNotification
 object: player];

[player beginGeneratingPlaybackNotifications];
}

```

Po zarejestrowaniu się na odpowiednie powiadomienia należy się upewnić, że są one prawidłowo wyrejestrowywane podczas czyszczenia pamięci i widoków — w przeciwnym razie aplikacja może się zawiesić lub dojdzie do innego nieoczekiwanego zachowania. Wywołanie metody `endGeneratingPlaybackNotifications` jest wykonywane w metodzie `viewDidUnload`.

```

-(void)viewWillDisappear:(BOOL)animated
{
    [[NSNotificationCenter defaultCenter] removeObserver: self
 name: MPMusicPlayerControllerNowPlayingItemDidChangeNotification
 object: player];

    [[NSNotificationCenter defaultCenter] removeObserver: self
 name: MPMusicPlayerControllerPlaybackStateDidChangeNotification
 object: player];

    [[NSNotificationCenter defaultCenter] removeObserver: self
 name: MPMusicPlayerControllerVolumeDidChangeNotification
 object: player];
    [player endGeneratingPlaybackNotifications];

    [super viewWillDisappear: animated];
}

```

Dodatkowo utworzymy zegar (`NSTimer`), który posłuży nam do odświeżania interfejsu użytkownika w miarę postępu w odtwarzaniu utworu. W przykładowej aplikacji obiekt klasy `NSTimer` nosi nazwę `playbackTimer`. Na razie wywołania zwrotne do obsługi powiadomień i zachowanie zegara pozostaną niedokończone. Omówimy je później w podrozdziale „Obsługa zmiany stanu”.

## Kontrolki użytkownika

Przykładowa aplikacja zawiera kilka przycisków zaprojektowanych do obsługi odtwarzanych utworów. Są to przycisk odtwarzania (ang. *play*), pauzy (ang. *pause*), pomiń (ang. *skip*) i poprzedni (ang. *previous*). Użytkownik będzie zmieniać pozycję w aktualnie odtwarzanym

utworze o pół minuty. Najpierw musimy zaimplementować metody do odtwarzania i wstrzymywania utworu. Obie funkcje to zwykły przełącznik — jeśli muzyka jest odtwarzana, przycisk wstrzyma odtwarzanie, a jeśli jest wstrzymana — powróci do odtwarzania. Kod odpowiedzialny za odświeżenie treści przycisku jest omówiony jako fragment wywołania zwrotnego dla powiadomienia zmiany stanu w podrozdziale „Obsługa zmiany stanu”.

```
- (IBAction)playButtonAction:(id)sender
{
    if ([player playbackState] == MPMusicPlaybackStatePlaying)
    {
        [player pause];
    }

    else
    {
        [player play];
    }
}
```

Użytkownik powinien móc pominąć aktualną ścieżkę lub przejść do poprzedniej. Osiągniemy to, wykonując dwa dodatkowe wywołania za pomocą obiektu player.

```
- (IBAction)previousButtonAction:(id)sender
{
    [player skipToPreviousItem];
}
- (IBAction)nextButtonAction:(id)sender
{
    [player skipToNextItem];
}
```

Użytkownicy będą mogli przejść o 30 sekund do przodu lub do tyłu. Jeśli użytkownik osiągnie koniec odtwarzanej ścieżki, poniższy kod spowoduje przejście do następnej. Podobnie jeśli zostanie osiągnięty moment za początkiem utworu, ścieżka zostanie ponownie odtworzona. Obie metody wykorzystują właściwość `currentTime` obiektu odtwarzacza. Właściwość ta może być używana do zmiany informacji na temat aktualnie odtwarzanego utworu.

```
- (IBAction)skipBack30Seconds:(id)sender
{
    int newPlayHead = player.currentTime - 30;

    if(newPlayHead < 0)
    {
        newPlayHead = 0;
    }

    player.currentTime = newPlayHead;
}

- (IBAction)skipForward30Seconds:(id)sender
```

```

{
    int newPlayHead = player.currentPlaybackTime + 30;

    if(newPlayHead > currentSongDuration)
    {
        [player skipToNextItem];
    }

    else
    {
        player.currentPlaybackTime = newPlayHead;
    }
}

```

Poza udostępnieniem standardowych kontrolerek w przykładowej aplikacji umożliwiamy także zmianę poziomu głośności audio. Obiekt odtwarzacza przyjmuje wartości od 0.0 (wyciszenie) do 1.0 (maksymalna głośność). Skoro korzystamy z kontrolki UISlider, wartość może być przekazana bezpośrednio do odtwarzacza.

```

- (IBAction)volumeSliderChanged:(id)sender
{
    [player setVolume:[volumeSlider value]];
}

```

## Obsługa zmiany stanu

Wspomniałem już wcześniej, że zostały zarejestrowane trzy powiadomienia, dla których otrzymamy wywołania zwrotne. Powiadomienia te pozwalają aplikacji na określanie stanu i zachowania klasy `MPMusicPlayerController`. Pierwsza metoda będzie wywoływana w momencie zmiany aktualnie odtwarzanego utworu. Metoda ta składa się z dwóch części. Najpierw aktualizujemy okładkę albumu, a następnie modyfikujemy etykiety opisujące artystę, tytuł piosenki i nazwę odtwarzanego albumu.

Każdy plik odtwarzany przy użyciu klasy `MPMusicPlayerController` jest reprezentowany za pomocą obiektu klasy `MPMediaItem`. Dostęp do obiektu jest możliwy przy użyciu metody `nowPlayingItem` tej klasy.

Okładka albumu jest tworzona za pomocą obiektu klasy `UIImage`. Domyślnie otrzymuje ona obrazek zastępczy, który będzie widoczny w sytuacji, gdy użytkownik nie dysponuje okładką albumu dla danego elementu typu `MPMediaItem`. Klasa `MPMediaItem` wykorzystuje właściwości w postaci klucz-wartość w celu przechowywania danych. Pełna lista właściwości została przedstawiona w tabeli 6.1. Następnie stworzymy obiekt typu `MPMediaItemArtwork` i przypisujemy do niego okładkę. Choć według dokumentacji w przypadku braku okładki powinniśmy otrzymać wartość `nil`, w praktyce tak się nie dzieje. Jest na to prosta rada — należy wczytać album do obiektu typu `UIImage` i sprawdzić otrzymaną wartość. Jeśli jest to wartość `nil`, możemy założyć, że nie dysponujemy okładką, w związku z czym zostanie załadowany obrazek zastępczy. Kod zawarty w przykładowej aplikacji będzie działał nawet wtedy, gdy klasa `MPMediaItemArtwork` zacznie zwracać wartość `nil` w przypadku braku okładki.

Tabela 6.1. Atrybuty dostępne w trakcie pracy z klasą MPMediaItem

<b>Atrybuty dostępne w klasie MPMediaItem</b>	<b>Dostępne do przeszukiwania predykatowego?</b>
MPMediaItemPropertyPersistentID	TAK
MPMediaItemPropertyAlbumPersistentID	TAK
MPMediaItemPropertyArtistPersistentID	TAK
MPMediaItemPropertyAlbumArtistPersistentID	TAK
MPMediaItemPropertyGenrePersistentID	TAK
MPMediaItemPropertyComposerPersistentID	TAK
MPMediaItemPropertyPodcastPersistentID	TAK
MPMediaItemPropertyMediaType	TAK
MPMediaItemPropertyTitle	TAK
MPMediaItemPropertyAlbumTitle	TAK
MPMediaItemPropertyArtist	TAK
MPMediaItemPropertyAlbumArtist	TAK
MPMediaItemPropertyGenre	TAK
MPMediaItemPropertyComposer	TAK
MPMediaItemPropertyPlaybackDuration	NIE
MPMediaItemPropertyAlbumTrackNumber	NIE
MPMediaItemPropertyAlbumTrackCount	NIE
MPMediaItemPropertyDiscNumber	NIE
MPMediaItemPropertyDiscCount	NIE
MPMediaItemPropertyArtwork	NIE
MPMediaItemPropertyLyrics	NIE
MPMediaItemPropertyIsCompilation	TAK
MPMediaItemPropertyReleaseDate	NIE
MPMediaItemPropertyBeatsPerMinute	NIE
MPMediaItemPropertyComments	NIE
MPMediaItemPropertyAssetURL	NIE
MPMediaItemPropertyIsCloudItem	TAK

Druga część metody `nowPlayingItemChanged`: zajmie się aktualizacją tytułu piosenki, informacji o wykonawcy i nazwy albumu (rysunek 6.1). Jeśli którakolwiek z tych właściwości zwróci `nil`, zostanie ustawiony obrazek zastępczy. Pełna lista dostępnych właściwości została przedstawiona w tabeli 6.1. Pamiętaj, że w przypadku podcastów będą dostępne

dotatkowe atrybuty, opisane przez Apple w dokumentacji dla klasy `MPMediaItem`. Dodatkowo w tabeli 6.1 określamy, czy klucz może być wykorzystany do przeszukiwania predykatowego w przypadku wyszukiwania z poziomu kodu obiektów typu `MPMediaItem`.

```
- (void) nowPlayingItemChanged: (id) notification
{
    MPMediaItem *currentItem = [player nowPlayingItem];

    UIImage *artworkImage = [UIImage imageNamed:@"noArt.png"];

    MPMediaItemArtwork *artwork = [currentItem valueForKey:@"MPMediaItemPropertyArtwork"];

    if (artwork)
    {
        artworkImage = [artwork imageWithSize: CGSizeMake (120,120)];

        if(artworkImage == nil)
        {
            artworkImage = [UIImage imageNamed:@"noArt.png"];
        }
    }
    [albumImageView setImage:artworkImage];

    NSString *titleString = [currentItem valueForKey:@"MPMediaItemPropertyTitle"];

    if (titleString)
    {
        songLabel.text = titleString;
    }

    else
    {
        songLabel.text = @"Nieznany utwór";
    }

    NSString *artistString = [currentItem valueForKey:@"MPMediaItemPropertyArtist"];

    if (artistString)
    {
        artistLabel.text = artistString;
    }

    else
    {
        artistLabel.text = @"Nieznany artysta";
    }

    NSString *albumString = [currentItem
valueForKey:@"MPMediaItemPropertyAlbumTitle];
```

```

    if (albumString)
    {
        recordLabel.text = albumString;
    }

    else
    {
        recordLabel.text = @"Nieznane nagranie";
    }
}

```

Monitorowanie stanu odtwarzacza jest niezwykle ważne, zwłaszcza że wartość ta może być modyfikowana poza aplikacją. W momencie zmiany stanu zostanie wywołana metoda `playbackStateChanged:`. Nowo tworzona zmienna `playbackState` służy do przechowywania stanu odtwarzacza. Metoda ta wykonuje wiele istotnych zadań — przede wszystkim aktualizuje stan przycisku odtwarzania, aby odzwierciedlić obecny stan. Ponadto metoda ta jest odpowiedzialna zarówno za tworzenie, jak i usuwanie obiektu klasy `NSTimer`, o którym wspominałem już wcześniej. W trakcie odtwarzania utworu zadanie zegara jest wykonywane co 0,3 sekundy. Do jego zadań należą aktualizacja etykiet związanych z odtwarzaniem oraz modyfikowanie stanu kontrolki `UIProgressIndicator` informującej użytkownika o momencie odtwarzania utworu, w którym obecnie się znajdujemy. Metodę, którą wykonuje zegar (`updateCurrentPlaybackTime`), omówimy w kolejnym podrozdziale.

Poza stanami przedstawionymi w przykładowej aplikacji możemy skorzystać z trzech dodatkowych. Pierwszy z nich — `MPMusicPlaybackStateInterrupted` — jest używany w momencie przerywania odtwarzania muzyki, np. w wyniku przychodzącego połączenia telefonicznego. Kolejne dwa stany — `MPMusicPlaybackStateSeekingForward` i `MPMusicPlaybackStateSeekingBackward` — są wykorzystywane do zgłoszenia, że stan odtwarzacza został przesunięty w przód lub w tył.

```

- (void) playbackStateChanged: (id) notification
{
    MPMusicPlaybackState playbackState = [player playbackState];

    if (playbackState == MPMusicPlaybackStatePaused)
    {
        [playButton setTitle:@"Odtwarzaj" forState: UIControlStateNormal];

        if ([playbackTimer isValid])
        {
            [playbackTimer invalidate];
        }
    }

    else if (playbackState == MPMusicPlaybackStatePlaying)
    {
        [playButton setTitle:@"Pauza" forState: UIControlStateNormal];

        playbackTimer = [NSTimer

```

```

        scheduledTimerWithTimeInterval:0.3
        target:self
        selector:@selector(updateCurrentPlaybackTime)
        userInfo:nil
        repeats:YES];
    }
    else if (playbackState == MPMusicPlaybackStateStopped)
    {
        [playButton setTitle:@"Odtwarzaj" forState:UIControlStateNormal];

        [player stop];

        if([playbackTimer isValid])
        {
            [playbackTimer invalidate];
        }
    }
}

```

W sytuacji gdy głośność uległa zmianie, niezwykle ważne jest odzwierciedlenie zmiany również w interfejsie użytkownika. Wystarczy obserwować powiadomienie `volumeChanged:`. Wewnątrz metody możemy pobrać aktualną głośność odtwarzacza i ustawić odpowiednio kontrolkę głośności.

```

- (void) volumeChanged: (id) notification
{
    [volumeSlider setValue:[player volume]];
}

```

## Czas trwania a zegary

Większość użytkowników zechce wiedzieć, w którym momencie trwania piosenki obecnie się znajdują lub ile czasu pozostało do jej końca. Przykładowa aplikacja zawiera dwie metody do generowania tych danych. Pierwsza z nich — `updateSongDuration` — jest wywoływana w momencie zmiany utworu lub uruchomienia aplikacji. Tworzymy w niej odwołanie do aktywnej ścieżki, a także pobieramy czas trwania przy użyciu klucza `playbackDuration`. Całkowita liczba godzin, minut i sekund jest obliczana bezpośrednio na podstawie tej wartości, a etykieta odpowiedzialna za wyświetlanie tych informacji jest przedstawiona tuż obok kontrolki `UIProgressIndicator`.

```

-(void)updateSongDuration;
{
    currentSongPlaybackTime = 0;

    currentSongDuration = [[[player nowPlayingItem] valueForKey:@"playbackDuration"] floatValue];

    int tHours = (currentSongDuration / 3600);
    int tMins = ((currentSongDuration / 60) - tHours*60);
    int tSecs = (currentSongDuration) - (tMins*60) - (tHours *3600);
}

```



```

songDurationLabel.text = [NSString stringWithFormat:@"%i:%02d:%02d",
↳tHours, tMins, tSecs];

currentTimeLabel.text = @"0:00:00";
}

```

Druga metoda, `updateCurrentPlaybackTime`, jest wykonywana co 0,3 sekundy przy użyciu zegara (`NSTimer`), który jest kontrolowany z poziomu poprzednio omówionej metody `playbackStateChanged:`. Do obliczenia godzin, minut oraz sekund korzystamy z tego samego algorytmu co w metodzie `updateSongDuration`. Wartość zmiennej `percentagePlayed` jest obliczana bezpośrednio na podstawie uzyskanego czasu trwania utworu. Korzystamy z niej do aktualizacji kontrolki `playbackProgressIndicator`. Biorąc pod uwagę, że czas przechowywany w zmiennej `currentPlaybackTime` jest podawany z dokładnością do jednej sekundy, teoretycznie nie musimy wywoływać jej zbyt często. Z drugiej strony im częściej będziemy wywoływać tę metodę, tym lepszą dokładność czasową uzyskamy.

```

-(void)updateCurrentPlaybackTime;
{
    currentSongPlaybackTime = player.currentPlaybackTime;

    int tHours = (currentSongPlaybackTime / 3600);
    int tMins = ((currentSongPlaybackTime / 60) - tHours*60);
    int tSecs = (currentSongPlaybackTime) - (tMins*60) - (tHours*3600);

    currentTimeLabel.text = [NSString stringWithFormat:@"%i:%02d:%02d",
↳tHours, tMins, tSecs];

    float percentagePlayed = currentSongPlaybackTime/currentSongDuration;

    [playbackProgressIndicator setProgress:percentagePlayed];
}

```

## Tryb losowy i powtarzanie

Do kolejnych funkcjonalności udostępnianych przez klasę `MPMusicPlayerController` zaliczamy możliwość powtórzenia odtwarzanego utworu i tryb losowy. Chociaż przykładowa aplikacja nie zawiera implementacji obu właściwości, dodanie jej nie jest zbyt skomplikowane:

```

player.repeatMode = MPMusicRepeatModeAll;
player.shuffleMode = MPMusicShuffleModeSongs;

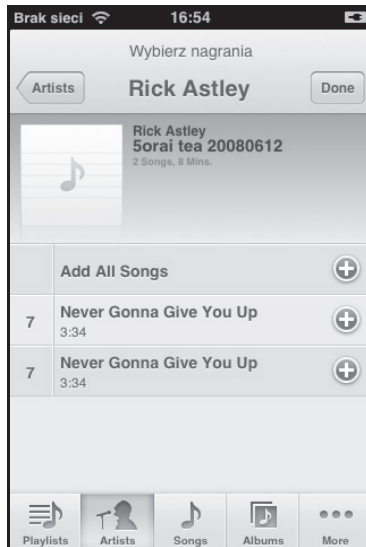
```

Dostępne tryby powtarzania to: `MPMusicRepeatModeDefault` (domyślny — zgodnie z wcześniejszym wyborem użytkownika), `MPMusicRepeatModeNone` (brak powtarzania), `MPMusicRepeatModeOne` (powtarzaj jedną) i `MPMusicRepeatModeAll` (powtarzaj całość).

Dostępne tryby losowania to: `MPMusicShuffleModeDefault` (domyślny — zgodnie z wcześniejszym wyborem użytkownika), `MPMusicShuffleModeOff` (brak losowania), `MPMusicShuffleModeSongs` (losuj piosenki) i `MPMusicShuffleModeAlbums` (losuj albumy).

## Media Picker — kontrolka wyboru multimediiów

Najprostszym sposobem na umożliwienie użytkownikowi wyboru odtwarzanej piosenki jest udostępnienie mu mechanizmu `MPMediaPickerController` (rysunek 6.2).



Rysunek 6.2. Wybór utworów przy użyciu kontrolera `MPMediaPickerController`. Elementy wybrane przez użytkownika są zaznaczone (są wyświetlane na szaro)

Klasa `MPMediaPickerController` pozwala użytkownikowi na przeglądanie swojego zbioru muzyki według artystów, piosenek, list odtwarzania i albumów. Zastosowanie klasy `MPMediaPickerController` wymaga zadeklarowania obsługi delegacji `MPMediaPickerControllerDelegate`, która zawiera dwie metody. Pierwsza z nich — `mediaPicker:didPickMediaItems` — jest wywoływana w momencie zakończenia wyboru utworów do odtworzenia. Wybrane utwory są zwracane w postaci obiektu `MPMediaItemCollection`, który możemy przekazać od razu do klasy `MPMusicPlayerController` jako parametr metody `setQueueWithItemCollection:`. Po ustawieniu nowej kolejki dla kontrolera `MPMusicPlayerController` możemy rozpocząć odtwarzanie nowo dodanych multimediiów. Kontroler `MPMediaPickerController` nie zostanie automatycznie usunięty po zakończeniu wyboru — konieczne jest jawne wywołanie metody `dismissViewControllerAnimated:completion:`.

```
- (void) mediaPicker: (MPMediaPickerController *) mediaPicker didPickMediaItems:
↳ (MPMediaItemCollection *) mediaItemCollection
{
    if (mediaItemCollection)
    {
```

```

        [player setQueueWithItemCollection: mediaItemCollection];
        [player play];
    }

    [self dismissViewControllerAnimated:YES completion:nil];
}

```

W przypadku anulowania lub zamknięcia kontrolera `MPMediaPickerController` bez dokonania wyboru zostanie wykonana metoda `mediaPickerDidCancel:`. Programista musi się pozbyć kontrolera właśnie w ramach tej metody:

```

- (void) mediaPickerDidCancel: (MPMediaPickerController *) mediaPicker
{
    [self dismissViewControllerAnimated:YES completion:nil];
}

```

Po zaimplementowaniu metod delegacji możemy utworzyć instancję klasy `MPMediaPickerController`. W momencie przeprowadzania alokacji i inicjalizacji obiektu tej klasy musimy przekazać parametr określający obsługiwane rodzaje multimediiów. Pełna lista dostępnych opcji została przedstawiona w tabeli 6.2. Zauważ, że każdy element może być skojarzony z wieloma rodzajami multimediiów. W tym miejscu możemy również skorzystać z kilku dodatkowych parametrów dla klasy `MPMediaPickerController`, np. możliwości zaznaczania wielu elementów czy wyświetlania znaku zachęty w trakcie zaznaczania (rysunek 6.2). Dodatkowy parametr typu logicznego pozwala określić, czy powinny być uwzględniane również elementy z usługi iCloud (domyślnie tak).

Tabela 6.2. Rodzaje mediów możliwe do zastosowania w momencie tworzenia kontrolera `MPMediaPickerController`

Stała	Opis
<code>MPMediaTypeMusic</code>	Dowolny element muzyczny
<code>MPMediaTypePodcast</code>	Podcast audio
<code>MPMediaTypeAudioBook</code>	Audiobook
<code>MPMediaTypeITunesU</code>	Element muzyczny skojarzony z usługą iTunes U
<code>MPMediaTypeAnyAudio</code>	Dowolny element dźwiękowy
<code>MPMediaTypeMovie</code>	Dowolny element zawierający film
<code>MPMediaTypeTVShow</code>	Dowolny element zawierający audycję telewizyjną
<code>MPMediaTypeVideoPodcast</code>	Dowolny element zawierający podcast wideo (nie należy go mylić z elementem typu <code>MPMediaTypePodcast</code> )
<code>MPMediaTypeMusicVideo</code>	Dowolny film zawierający nagranie muzyczne
<code>MPMediaTypeVideoITunesU</code>	Dowolny element wideo skojarzony z usługą iTunes U (nie należy go mylić z elementem typu <code>MPMediaTypeAudioITunesU</code> )
<code>MPMediaTypeAnyVideo</code>	Dowolny element wideo
<code>MPMediaTypeAny</code>	Dowolny element zawierający zarówno audio, jak i wideo

```

- (IBAction)mediaPickerButtonAction:(id)sender
{
    MPMediaPickerController *mediaPicker = [[MPMediaPickerController alloc]
    initWithMediaTypes:MPMediaTypeAny];

    mediaPicker.delegate = self;
    mediaPicker.allowsPickingMultipleItems = YES;
    mediaPicker.prompt = @"Wybierz utwory do odtworzenia";
    [self presentViewController:mediaPicker animated:YES completion:nil];

    [mediaPicker release];
}

```

Na tym kończymy opis czynności wymaganych do wybrania muzyki do odtwarzania przy użyciu klasy `MPMediaPickerController`. Czasami może się jednak zdarzyć, że konieczne będzie stworzenie własnego interfejsu użytkownika do tego celu, a nawet umożliwienie wyboru piosenek bez pomocy interfejsu! Tyimi kwestiami zajmujemy się w kolejnym podrozdziale.

## Ręczny wybór piosenek

Mogą zdarzyć się sytuacje, w których konieczne będzie opracowanie bardziej szczegółowego interfejsu do wyboru muzyki przez użytkownika. Może się to wiązać np. z utworzeniem innego zestawu kontrolki do wyboru pojedynczych utworów lub automatycznym wyszukiwaniem artysty czy albumu. W tym podrozdziale omówimy czynności niezbędne do zapewnienia możliwości wyboru piosenek w dowolny sposób.

Pobranie piosenek bez użycia klasy `MPMediaPickerController` wymaga utworzenia obiektu klasy `MPMediaQuery`. Obiekt taki funkcjonuje jako magazyn przechowujący określoną liczbę elementów typu `MPMediaItem`, z których każdy reprezentuje pojedynczy utwór lub ścieżkę audio.

Przykładowa aplikacja udostępnia dwie metody, które korzystają z klasy `MPMediaQuery`. Pierwsza z nich — `playRandomSongAction` — wybierze losowo piosenkę z biblioteki użytkownika i odtworzy ją, korzystając z istniejącego kontrolera `MPMusicPlayerController`. Znalezienie muzyki z poziomu kodu wiąże się z utworzeniem nowego obiektu klasy `MPMediaQuery`.

## Odtwarzanie losowo wybranej piosenki

Jeśli nie określimy żadnego predykatu (warunku przeszukiwania), obiekt klasy `MPMediaQuery` znajdzie wszystkie elementy zawarte w ramach biblioteki muzyki. Dane znajdują się w nowej tablicy (`NSArray`), z której skorzystamy za pomocą metody elementu w klasie `MPMediaQuery`. Każdy element jest reprezentowany oczywiście przy użyciu obiektu klasy `MPMediaItem`. Funkcjonalność losowego wybierania piosenki pozwoli na odtworzenie jednej losowo wybranej piosenki. Jeśli nie uda się znaleźć żadnych utworów, obiekt typu `UIAlert` zostanie zaprezentowany użytkownikowi. W przeciwnym razie wybór nastąpi losowo spośród wszystkich znalezionych utworów.

Po znalezieniu jednego lub większej liczby elementów typu `MPMediaItem` tworzymy obiekt typu `MPMediaItemCollection`, przekazując do niego tablicę elementów typu `MPMediaItem`. Kolekcja posłuży jako lista odtwarzania dla kontrolera `MPMusicPlayerController`. Po utworzeniu kolekcji obiekt jest przekazywany do odtwarzacza przy użyciu metody `setQueueWithItemCollection`. W tym momencie odtwarzacz wie, które piosenki zostaną odtworzone przez użytkownika, dlatego też zostanie rozpoczęte odtwarzanie piosenek jedna po drugiej, w takiej kolejności, w jakiej znajdują się one w tablicy przekazanej w momencie tworzenia kolekcji typu `MPMediaItemCollection`.

```
- (IBAction)playRandomSongAction:(id)sender
{
    MPMediaItem *itemToPlay = nil;
    MPMediaQuery *allSongQuery = [MPMediaQuery songsQuery];
    NSArray *allTracks = [allSongQuery items];

    if([allTracks count] == 0)
    {
        UIAlertView *alert = [[UIAlertView alloc]
            initWithTitle:@"Błąd"
            message:@"Nie znaleziono utworów!"
            delegate:nil
            cancelButtonTitle:@"Anuluj"
            otherButtonTitles:nil];

        [alert show];
        [alert release];
        return;
    }

    if ([allTracks count] < 2)
    {
        itemToPlay = [allTracks lastObject];
    }

    int trackNumber = arc4random() % [allTracks count];
    itemToPlay = [allTracks objectAtIndex:trackNumber];

    MPMediaItemCollection * collection = [[MPMediaItemCollection alloc]
        initWithItems:[NSArray arrayWithObject:itemToPlay]];

    [player setQueueWithItemCollection:collection];
    [collection release];

    [player play];

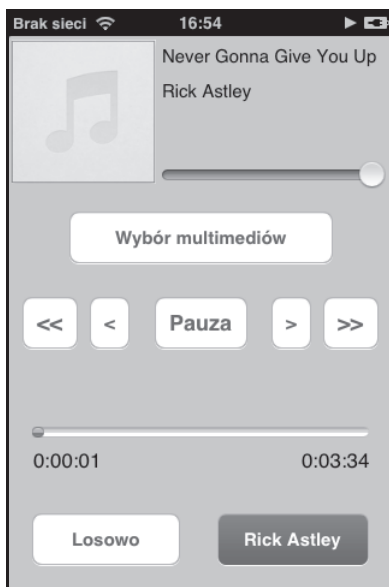
    [self updateSongDuration];
    [self updateCurrentPlaybackTime];
}
```

**Uwaga**

Funkcja `arc4random()` należy do standardowej biblioteki języka C. Można z niej korzystać do generowania liczb pseudolosowych w projektach tworzonych w języku Objective-C. W przeciwieństwie do większości generatorów liczb pseudolosowych funkcja ta otrzymuje ziarno automatycznie w momencie pierwszego użycia.

**Wyszukiwanie piosenek w oparciu o predykaty**

Odtwarzanie losowo wybranych piosenek może się znudzić. W takiej sytuacji przydałaby się możliwość przeszukiwania biblioteki muzyki. Funkcjonalność tę możemy osiągnąć przy użyciu predykatów. W poniższym przykładzie korzystamy z predykatu w celu znalezienia elementów, w których atrybut `artist` ma wartość "Rick Astley" (rysunek 6.3).



Rysunek 6.3. Odtwarzamy tylko piosenki Ricka Astleya dzięki przeszukiwaniu opartemu na predykatach

Przedstawiona metoda działa podobnie do losowego wybierania piosenek, z tą różnicą, że korzystamy z metody `addFilterPredicate` w celu dodania filtru do klasy `MPMediaQuery`. Co więcej, wyniki nie są ograniczane do jednego elementu — odtwarzacz otrzyma tablicę wszystkich pasujących utworów. Pełna lista dostępnych predykatów została przedstawiona w tabeli 6.1 w podrozdziale „Obsługa zmiany stanu”. Aby skorzystać z wielu predykatów jednocześnie, wystarczy wykonać kolejne wywołania metody `addFilterPredicate` w klasie `MPMediaQuery`.

```

- (IBAction)playDylan:(id)sender
{
    MPMediaPropertyPredicate *artistNamePredicate = [MPMediaPropertyPredicate
↳predicateWithValue:@"Rick Astley" forProperty:MPMediaItemPropertyArtist];

    MPMediaQuery *artistQuery = [[MPMediaQuery alloc] init];

    [artistQuery addFilterPredicate:artistNamePredicate];

    NSArray *tracks = [artistQuery items];

    if([tracks count] == 0)
    {
        UIAlertView *alert = [[UIAlertView alloc]
            initWithTitle:@"Błąd"
            message:@"Nie znaleziono utworów!"
            delegate:nil
            cancelButtonTitle:@"Anuluj"
            otherButtonTitles:nil];

        [alert show];
        [alert release];

        return;
    }

    MPMediaItemCollection *collection = [[MPMediaItemCollection alloc]
↳initWithItems:tracks];

    [player setQueueWithItemCollection:collection];
    [collection release];

    [player play];

    [self updateSongDuration];
    [self updateCurrentPlaybackTime];
}

```

## Podsumowanie

W tym rozdziale omówiliśmy mechanizmy związane z obsługą biblioteki muzyki użytkownika. Najpierw stworzyliśmy silnik odtwarzania, który pozwala użytkownikowi kontrolować proces odtwarzania, np. wstrzymywanie, powracanie, kontrolę głośności i przechodzenie pomiędzy kolejnymi utworami. W kolejnych dwóch podrozdziałach zajęliśmy się dostępem do samej biblioteki i wyborem z niej konkretnych utworów. Zademonstrowaliśmy możliwości kontrolki wyboru multimediów za pomocą wbudowanego interfejsu. Omówiliśmy także możliwości ręcznego przeprowadzenia procesu wyboru utworów z wykorzystaniem predykatów.

Przykładowa aplikacja stanowi przykład w pełni funkcjonalnego, ale jednocześnie stosunkowo prostego odtwarzacza muzyki dla systemu iOS. Wiedza przedstawiona w tym rozdziale może się stać podstawą do stworzenia jeszcze bardziej rozbudowanych odtwarzaczy, a także do opracowania mechanizmów odtwarzania muzyki w tle w dowolnej aplikacji.

## Ćwiczenia

1. Dodaj znacznik do kontrolki postępu odtwarzania, który umożliwi zmianę aktualnego momentu odtwarzania utworu, pozwalając na pełną kontrolę użytkownika nad odtwarzaniem.
2. Dodaj funkcjonalność, która pozwoli użytkownikowi na wprowadzenie nazwy piosenki lub artysty, a następnie przedstawi listę pasujących elementów, po czym umożliwi ich odtworzenie.



# Skorowidz

## A

- adnotacje, 56
- adres URL serwera, 223
- AirPrint, 245
- aktualizacja kuponu, 503, 506
- aktywności wykonywane w tle, 352
- animacja, 31
- animacje widoku kolekcji, 424–427
- anulowanie
  - drukowania, 253
  - operacji, 369
- aplikacja
  - AssetNavigator, 457
  - Czasochłonne zadania, 359
  - do drukowania, 246
  - FavoritePlaces, 43, 58, 64, 68
  - Instruments, 520
  - Mapy, 76
  - Message Board, 170
  - Messages, 230
  - Muzyka, 152
  - MyMovies, 269
  - MyNotes, 182, 195
  - Obrazkowe zabawy, 387
  - Odtwarzacz, 151
  - Passbook, 475, 507
  - Pęk kluczy, 378
  - PhotoGallery, 407
  - Print Center, 252
  - ShoutOut, 202
  - SocialNetworking, 322
  - Test kuponów, 476, 499

- Ustawienia, 45, 134
- Whack-a-Cac, 81
- Zabawa gestami, 444
- Zadania w tle, 348
- APNs, 227
- ARC, Automatic Reference Counting, 138, 522
- archiwa kluczowane, 258
- argument category, 100
- asystent certyfikatów, 207, 492, 493
- atrybut Point Value, 113
- atrybuty
  - filtra, 397
  - MPMediaItem, 157
  - tekstowe, 431, 432
- automatyczna aktualizacja, 506
- automatyczne zliczanie referencji, 299
- autoryzacja, 46

## B

- biblioteka
  - Asset Library, 474
  - Core Bluetooth, 230
  - Core Data, 57, 71, 266, 269
  - Core Image, 387, 395
  - Core Location, 53
  - Core Text, 429
  - Enterprise Object Framework, 257
  - Game Kit, 229
  - libsqlite, 259
  - PassKit, 476
  - PassKit.framework, 497

biblioteka  
 Quartz, 387  
 TextKit, 430  
 UIKit Dynamics, 29, 41  
 zasobów, Assets Library, 391, 458

biblioteki muzyki, 151

bilet podróży, 478

blok enumerateAssetGroupsBlock, 460

bloki, blocks, 299, 307, 311

Bluetooth, 229

błąd  
 asercji, 419  
 ładowania, 497  
 uwierzytelniania, 96  
 użyteczności, 86

## C

centrum drukowania, Print Center, 251

certyfiikat  
 iOS, 206  
 podpisywania kuponów, 491  
 SSL, 206, 208

chmura, 181

Core Data, 257, 269

Core Location, 43

czas  
 odtwarzania, 160  
 rozgrywki, 128

## D

dane  
 JSON, 174  
 obiektu, 283

debugger  
 GDB, 515  
 LLDB, 515

debugowanie, 509, 511

deklarowanie  
 bloków, 308  
 właściwości, 313  
 związków, 276

delegacja  
 klasy, 48  
 MKMapViewDelegate, 56  
 UICollisionBehaviorDelegate, 34  
 UIPrintInteractionControllerDelegate,  
 253  
 widoku kolekcji, 413

detektor  
 stuknięć, 445  
 szczygnięć, 446

detektory gestów, 443, 450

dodawanie  
 adnotacji, 57  
 atrybutów, 275  
 filtra, 396, 398  
 haków osiągnięć, 119  
 kuponów, 501  
 obiektów zarządzanych, 293  
 obsługi książki adresowej, 135  
 osiągnięcia, 112  
 osiągnięcie do gry, 123  
 trasy, 76  
 ulubionego miejsca, 65  
 wyników, 102

dostęp do  
 grup zasobów, 458  
 klasy, 459  
 obszaru debugowania, 511  
 osi czasu, 336  
 pęku kluczy, 492, 494  
 położenia urządzenia, 44  
 serwera, 171  
 SQLite, 258  
 tablicy obiektów, 130  
 właściwości, 315

dostępność usług lokalizacyjnych, 48

dostosowywanie  
 elementów, 30  
 punktów przerwania, 513  
 widoku kolekcji, 415

drukarka bezprzewodowa, 245

drukowanie, 245, 250  
 dokumentu HTML, 254  
 plików PDF, 255  
 tekstu, 247

drzewo wywołań, 525  
 dymek, 232  
 dynamiczne  
   połączenie, 35  
   wykrywanie hiperłączy, 430, 433

## E

edycja  
   kontaktów, 144  
   obiektu zarządzanego, 294  
 edytor  
   listy kontaktów, 134  
   modelu danych, 261, 274  
   tekstowy, 246  
 edytowanie obiektów zarządzanych, 293  
 efekt sprężynowania, 36  
 efekty, 30  
 ekran wykrzyknień, 220  
 elementy dynamiczne, 39  
 encja, 260, 275, 276  
 etykiety, 139

## F

Facebook, 321, 341  
 filtry biblioteki Core Image, 395, 397  
 format  
   JSON, 169, 170, 176, 483  
   PDF 417, 485  
   PNG, 483  
 framework  
   AddressBook.framework, 135  
   AddressBookUI.framework, 135  
   Core Data, 257  
   Core Image, 395  
   Core Location, 43  
   MapKit, 43  
   społecznościowy, 321  
 funkcja  
   main, 520  
   scaffold, 219  
 funkcjonalności Core Data, 257

## G

galeria zdjęć, 407  
 Game Kit, 229  
 geofencing, 79  
 geokoder, 67  
 geokodowanie, 43, 65, 67  
 gesty, 443  
   stuknięcie, 445  
   szczypnięcie, 446  
 głośność, 160  
 główny wątek, 361  
 GPS, 49  
 gra Whack-a-Cac, 81, 102  
 graf modeli, 261  
 Grand Central Dispatch, 359  
 grawitacja, 32  
 grupowanie osiągnięć, 127  
 grupy zasobów, 458, 463, 469, 473

## H

HIG, Human Interface Guidelines, 17

## I

iCloud, *Patrz* usługa iCloud  
 IDE, Integrated Development Environment, 510  
 identyfikacja  
   kodu kreskowego, 485  
   kuponu, 484  
 identyfikator  
   komórki, 410  
   obiektu, 282  
   tablic, 92  
   typu kuponu, 489  
   zadania w tle, 350  
   zarządzanego obiektu, 73  
 implementacja protokołu MKAnnotation, 57  
 informacje  
   o drukowaniu, 248, 253  
   o grupach, 461  
   o kuponie, 484

- informacje
  - o mediach, 393
  - o miejscu, 68
  - o osiągnięciach, 114
  - o ulubionym miejscu, 74
  - o zmianach położenia, 48, 50
- inicjalizacja
  - obrazka, 400
  - usługi iCloud, 184
- inspektor tożsamości, identity inspector, 409
- instrukcja import, 44
- instrumenty, 517, 522, 526
- interakcja
  - z kuponami, 497
  - z usługą iCloud, 188
- Interface Builder, 416
- interfejs
  - graficzny książki adresowej, 141
  - instrumentów, 518, 519
- interpretacja danych położenia, 50

## J

- JSON, 169

## K

- karta podarunkowa, 480
- klasa
  - ABAddressBookRef, 135
  - ABNewPersonViewController, 145
  - ABPersonViewController, 144
  - ABRecordRef, 138
  - ALAsset, 458
  - ALAssetRepresentation, 458
  - ALAssetsGroup, 458
  - ALAssetsLibrary, 458, 459
  - CLLocation, 50, 69
  - CLLocationCoordinate2D, 50, 57
  - CLLocationCoordinate2DMake, 57
  - CLLocationDistance, 51
  - CLLocationManager, 48, 49
  - CLPlacemark, 68
  - GKAchievement, 116
  - GKAchievementDescription, 130

- ICFAssetGroupViewController, 464, 471
- ICFConflictVersionViewController, 194
- ICFFavoritePlace, 57
- ICFFavoritePlaceViewController, 65, 71, 75
- ICFFilterCategoriesViewController, 395
- ICFFriendChooserViewController, 280
- ICFGameCenterManager, 99, 115, 129, 130
- ICFGameViewController, 83
- ICFLocationManager, 46, 48
- ICFMainViewController, 54, 224
- ICFMovieListViewController, 285, 288
- ICFNewMessageViewController, 175
- ICFPerformBackgroundViewController, 363
- ICFViewController, 82, 102, 104
- Keychain, 377
- KeychainItemWrapper, 379, 381
- kSecClassGenericPasswora, 381
- MKAnnotationView, 61
- MKDirectionsRequest, 76
- MKMapItem, 75
- MKPinAnnotationView, 61
- MKPointAnnotation, 57
- MPMediaItem, 157
- MPMediaPickerController, 162, 164
- MPMediaQuery, 164, 166
- MPMusicPlayerController, 153, 156, 162
- NSEntityDescription, 278, 293
- NSFileManager, 184
- NSFileVersion, 193
- NSJSONSerialization, 177
- NSLayoutManager, 430, 432
- NSManagedObject, 57, 259
- NSMetadataQuery, 188, 189
- NSNotificationCenter, 153
- NSTextContainer, 433
- NSTextStorage, 431
- NSTimer, 129
- NSUserDefaults, 128
- PHGSectionHeader, 412
- SLComposeViewController, 322, 324
- UICollectionViewCell, 410

- UICollectionViewController, 409
- UIDocument, 185, 186, 187
- UIDynamicAnimator, 41
- UIDynamicAnimatorDelegate, 41
- UIDynamicItem, 40
- UIGestureRecognizer, 454
- UIImage, 388
- UIImagePickerController, 391, 392
- klucz, 377
  - PIN, 379
  - prywatny, 494
- klucze atrybutów, 381
- kod kreskowy, 485
- kody błędów pęków kluczy, 385
- kolejka, 361
  - dyspozytora, 371–373
  - GCD, 361
- kolekcja typu MPMediaItemCollection, 165
- kolizje, 33
- komponent Media Picker, 151
- komunikacja Bluetooth, 230
- komunikat
  - o braku uprawnień, 46
  - o udzielenie zgody, 44
- konfiguracja
  - detektora twarzy, 403
  - domyślnych danych, 278, 279
  - konta programisty, 182
  - MKMapKit, 54
  - pęków kluczy, 385
  - pęku kluczy, 378
  - Rails, 218
  - ShoutOut, 203
  - tablicy wyników, 91, 92
  - widoku adnotacji, 62
  - widoku kolekcji, 408
  - źródła danych, 411
- kontekst obiektów zarządzanych, 266
- konto programisty iOS, 182
- kontroler
  - ABNewPersonViewController, 145
  - ABPeoplePickerNavigationController, 142
  - ABPersonViewController, 144
  - GKAchievementViewController, 114
  - GKLeaderboardViewController, 104
  - ICFChatViewController, 236
  - ICFMovieEditViewController, 293
  - ICFViewController, 395
  - MPMediaPickerController, 162, 163
  - NSFetchedResultsController, 270
  - PKAddPassesViewController, 502
  - UINavigationController, 430
- kontrolery
  - dodawania kuponów, 501
  - pobranych wyników, 287
  - urządzeń, 222
  - widoków, 194
  - wykrzykników, 222
- kontrolka
  - Image Picker, 391
  - Media Picker, 162
  - Peer Picker, 234
  - UITableView, 174
  - wyboru kontaktów, 142
- konwersja projektu, 303
- kończenie zadania w tle, 351
- koordynator trwałego magazynu, 265, 272
- kopiowanie książki adresowej, 135
- krzywe Béziera, 430
- książka adresowa
  - etykiety, 139
  - interfejs graficzny, 141
  - obsługa, 133, 135
  - obsługa adresów, 140
  - odczytywanie wielowartości, 139
  - ograniczenia, 134
  - wczytywanie danych, 137
- kupon, 477
  - bilet podróży, 478
  - karta podarunkowa, 480
  - ogólny, 479
  - wydarzenie, 478
  - zniżkowy, 478
- kwalifikatory ARC, 307

**L**

## lista

- efektów, 30
- notatek, 190
- właściwości, 258
- wyników, 91

literały, literals, 299

logowanie do Facebooka, 323

losowanie utworu, 161

**Ł**

ładowanie kuponów, 497

łączenie filtrów, 401

**M**

magazyn trwałe, 266

manifest, 495

mapa, 53

MapKit, 43

menedżer

- Game Center, 94, 115
- lokalizacji, 49, 72
- NSLayoutManager, 431

metoda

- ABAddressBookAddRecord, 146
- ABAddressBookCopyArrayOfAllPeople, 136
- ABAddressBookCopyArrayOfAllPeople
  - ↳ InSource, 136
- ABAddressBookCopyLocalizedLabel, 139
- ABAddressBookCreate, 136
- ABAddressBookGetPersonCount, 136
- ABAddressBookSave, 146
- ABMultiValueCopyLabelAtIndex, 139
- ABMultiValueCopyValueAtIndex, 139
- ABMultiValueCreateMutable, 147
- acceptConnectionFromPeer:, 243
- addAttribute:range:, 438
- addBehavior, 32, 41
- addFilterPredicate, 166
- authenticate, 98

cactusHit:, 85, 124

cactusMissed:, 86

calculateDirectionsWithCompletion

- ↳ Handler:, 76

cancelButtonTouched:, 296

cellForRowAtIndexPath:, 231, 525

collectionView:umberOfItemsInSection:, 411

collectionViewContentSize, 420

configureView, 197

connect:, 235

contentsForType:error:, 186

dismissViewControllerAnimated:

- ↳ completion:, 162

displayNewScore:, 85, 87, 125

endGeneratingPlaybackNotifications, 154

enumerateAssetsForGroup:, 466

enumerateSubstringsInRange:, 438

finalizeCollectionViewUpdates, 428

finalLayoutAttributesForDisappearing

- ↳ ItemAtIndexPath:, 428

GameCenterManager reportScore:, 103

getCellIdentifierForAttributeType:, 399

getDirectionsButtonTouch:, 75

heightForRowAtIndexPath:, 233

init, 48

iPodMusicPlayer, 153

issueChallengeToPlayers:withMessage:, 106

JSONObjectWithData:options:error:, 174

layoutAttributesForDecoration

- ↳ ViewOfKind:atIndexPath:, 421

layoutAttributesForElementsInRect:, 420

layoutAttributesForItemAtIndexPath:, 420

layoutAttributesForSupplementary

- ↳ ViewOfKind:atIndexPath:, 420

loadAchievementsWithCompletion

- ↳ Handler, 115, 116

locationManager:didEnterRegion:, 72

locationManager:didExitRegion:, 72, 73

locationManager:didFailWithError:, 50

- locationManager:didUpdateLocations:, 49, 53
  - locationServicesEnabled, 48
  - mapView:annotationView:callout
    - ↳AccessoryControlTapped:, 61
  - mapView:annotationView:didChange
    - ↳DragStatefromOldState:, 62
  - mapView:annotationView:viewFor
    - ↳Annotation, 59
  - mapView:annotationView:viewFor
    - ↳Overlay:, 78
  - nowPlayingItem, 156
  - NSOperationQueue, 360
  - numberOfAssets, 462
  - objectAtIndex:, 289
  - openInMapsWithlaunchOptions:, 75
  - peoplePickerNavigationController:should
    - ↳ContinueAfterSelectingPerson:, 143
  - performLongRunningTaskForIteration:, 364
  - performSelectorInBackground:withObject, 360
  - personViewController:shouldPerform
    - ↳DefaultActionForPerson:property:
      - ↳identifier, 145
  - play5MinTick, 129
  - playRandomSongAction:, 164
  - prepareLayout, 421
  - presentAnimated:completionHandler:, 250
  - processFiles:, 190
  - receiveData:fromPeer:inSession:context, 240
  - registerForRemoteNotificationTypes:, 214
  - registerMediaPlayerNotifications, 153
  - reportAchievement:withPercentage
    - ↳Complete:, 117, 120
  - reportScore:forCategory, 99
  - reportScoreWithCompletionHandler, 101
  - resolveConflictTapped:, 193
  - reverseGeocodeDraggedAnnotation:
    - ↳forAnnotationView:, 69
  - saveButtonTouched:, 176
  - saveToURL:forSaveOperation:completion
    - ↳Handler:, 188
  - sendMessage:, 239
  - setDamping, 36
  - setFrequency, 36
  - setImage:, 62
  - setQueueWithItemCollection, 165
  - shouldInvalidateLayoutForBounds
    - ↳Change:, 421
  - showInfo:, 224
  - showPassTouched, 505
  - spawnCactus, 83
  - tableView:didSelectRowAtIndexPath:, 290
  - textFieldShouldReturn:, 225
  - UIImagePickerController, 471
  - updateCurrentPlaybackTime, 161
  - updateLife, 86, 103
  - updateMapAnnotations, 58, 63
  - updatePassTouched, 503
  - updateSongDuration, 160
  - URLForUbiquityContainerIdentifier, 184
  - viewDidLoad, 82, 136, 153, 445, 464
  - viewWillDisappear, 191
  - zoomMapToFitAnnotations, 55
  - metody
    - delegacji
      - UIPrintInteractionControllerDelegate, 253
    - źródła danych, 411
  - migracje modeli, 262
  - model obiektu zarządzanego, 260, 273
  - monitorowanie
    - obszaru, 71
    - położenia, 52
    - stanu odtwarzacza, 159
    - zmian, 72
- ## N
- NARC, New, Alloc, Retain, Copy, 138
  - narzędzia, 517

## narzędzie

- Allocations, 519, 523, 525
- Interface Builder, 411
- Keychain Access, 207, 492
- Leaks, 523

nawiązywanie połączenia, 243

nazwa partnera, 242

notacja kropkowa, 315

**O**

## obiekt

- chatObjectArray, 232
- geokodera, 67
- GKAchievement, 127
- GKScore, 100
- MKRouteStep, 76
- NSNumber, 300
- UIAlertView, 330
- UIImage, 389

obiekty zarządzane, 259, 263, 266

obrazki, 387

obrót, 449

## obsługa

- adresów, 140
- bibliotek muzyki, 151
- bibliotek zdjęć, 457
- detektorów gestów, 444
- dostosowywania elementów, 30
- dotknięć, 451
- formatu JSON, 169
- gestów, 443
- informacji zwrotnej, 227
- książki adresowej, 133, 135
- kuponów, 498
- miejsc i map, 43
- odtworacza, 154
- odwrotnego geokodowania, 71
- przeciągalnych adnotacji, 56
- strumienia zdjęć, 472
- stuknięć, 56
- typów dynamicznych, 441
- urządzeń, 219
- wygaśnięcia, 350

wykrzyknień, 219

zdjęć, 457

zmiany stanu, 156

odpychanie, 37

odtworacz muzyki, 152, 164

odtworzenie muzyki w tle, 354

odwrotna notacja DNS, 92, 112, 490

odwrotne geokodowanie, 62, 65–70

odwzorowanie Merkatora, 53

ograniczenia komunikacji, 229

określanie granic, 72

opakowania, boxed expressions, 302

opcje

drukowania, 251

formatowania, 93

nowego projektu, 271

operacje

szeregowe, 367

własne, 369

współbieżne, 366

osiągnięcia, achievements, 111, 123

częściowe, 125

nieukończone, 124

ukończone, 124

z wykorzystaniem zegarów, 128

ostrzeżenie o błędzie, 47

otrzymywanie

danych, 240

powiadomień, 216

**P**

pamięć, 311

pamięć podręczna osiągnięć, 115

parametry metod, 310

parsowanie, 50

parsowanie JSON, 169, 173

PassKit, 475

Peer Picker, 235

pęk kluczy, 377

planowanie powiadomień, 215

plik

boarding\_pass.pkpass, 496

DMNS.gpx, 52



- ICFGameViewController.m, 87
  - ICFViewController.m, 82
  - manifest.json, 495
  - pass.json, 506
  - RootViewController.h, 135
  - ShoutOut-Prefix.pch, 223
  - pliki
    - GPX, 52
    - nagłówkowe, 135
  - pobieranie
    - danych, 171
    - informacji, 48
    - obiektu, 282
    - PIN-u, 380
    - wyników, 288
  - podgląd wydruku, 252
  - podklasy
    - klasy UIDocument, 186
    - obiektów zarządzanych, 277
  - podmiana metod, 299, 317
  - podpisywanie
    - kodu, 213
    - paczki, 489, 495
  - podświetlanie zależne od treści, 436, 439
  - poła kuponu, 486
  - polecenia debuggera, 516
  - położenie, 44, 50–52
  - portal poświadczeń, 205
  - postęp osiągnięcia, 119, 127
  - poświadczenia, 206–212, 490–493, 498
  - powiadomienia, 120, 201–227
    - lokalne, 216
    - odtwarzania, 153
    - zdalne, 206, 217, 226
  - powiadomienie typu push, 507
  - powiązanie, attachment, 35
  - powtarzanie utworu, 161
  - predykat, 166, 284
  - profiler czasowy, 520, 521
  - programistyczny profil poświadczeń, 209
  - projektowanie kuponu, 477
  - protokół MKAnnotation, 57
  - prywatność użytkowników, 44
  - przechowywanie danych, 258
  - przechwytywanie stanu, 309
  - przeciągalne widoki adnotacji, 62
  - przeglądarka, 134
  - przeglądarka kontaktów, 144
  - przekształcanie danych, 176
  - przetwarzanie
    - odpowiedzi, 172
    - rysów twarzy, 404
  - przyciąganie, 37
  - punkty
    - odniesienia, waypoints, 52
    - przerwania, breakpoints, 512–514
    - zaczepienia, anchor points, 35
- ## R
- reagowanie na zmiany, 290
  - reduks, 451
  - rejestracja
    - identyfikatora aplikacji, 203, 204
    - powiadomień odtwarzania, 153
  - renderowanie
    - obrazka, 401
    - tekstu, 442
  - resetowanie
    - elementu pęku kluczy, 384
    - osiągnięć, 129
  - rodzaje
    - aktywności wykonywanych w tle, 353
    - detektorów gestów, 443
    - kuponów, 477
    - map, 55
    - mediów, 163
  - rozmiar
    - książki adresowej, 136
    - obrazka, 394
  - rozpoznawanie gestów, 443
  - rozszerzenie .json, 172
  - rozwiązywanie konfliktów, 195
  - ruch przyciągania, snapping motion, 37

**S**

schemat aplikacji FavoritePlaces, 53  
 serializacja, 177  
 serwer  
   powiadomień zdalnych, 217  
   Ruby on Rails, 218  
 serwis iTunes Connect, 111  
 serwisy społecznościowe, 321  
 silnik  
   odtworzenia, 152  
   stanów, 82  
 skalowanie, 449  
 składowe adresu, 141  
 słownik, 382  
   achievementDictionary, 117  
   NSDictionary, 148, 301, 461  
   NSMutableDictionary, 115  
 sortowanie obiektów, 263  
 sprawdzanie  
   dostępności usług, 48  
   uprawnień, 460  
 sprzężowanie, spring, 36  
 SQLite, 258  
 stan odtwarzacza, 159  
 stany wyzwania, 107, 122  
 stosowanie  
   ARC, 303, 305  
   bloków, 308  
   Bluetooth, 230  
   Core Data, 258  
   formatu JSON, 170  
   klasy SLComposeViewController, 324  
   książki adresowej, 134  
   pęku kluczy, 377  
   predykatów, 284  
 strumień zdjęć, 457, 472  
 stuknięcie, 445  
 style  
   czcionki, 442  
   tabeli, 261  
 symboliczne punkty przerwania, 514  
 symulacja  
   aktualizacji kuponu, 503  
   położenia, 52, 54

symulator drukowania, 246, 251  
 synchronizacja słowników, 197  
 syntetyzacja właściwości, 315  
 szablon Master-Detail, 270  
 szablony nowego projektu, 271  
 szczygnięcie, 446  
 szybkie wyliczenia, 299, 316

**Ś**

ścieżki wyłączające, 430, 435  
 śledzenie wycieków, 524  
 środowisko Core Data, 265, 272

**T**

tablica  
   assetArray, 468  
   NSArray, 136, 301  
   placemarks, 68  
 tablice  
   ogłoszeń, 170, 171  
   wyników, 81, 91, 108  
 technologia  
   AirPrint, 245  
   ARC, 138  
   Core Data, 270  
 testowanie  
   AirPrint, 247  
   dynamicznego kodu, 29  
   kuponu, 496  
   lokalizacji, 52  
   osiągnięć, 130  
 TextKit, 429  
 transfer danych, 238  
 trasa, 77  
 tryb grafu, 275  
 tryby  
   sesji, 244  
   transferu danych, 238  
 Twitter, 321, 337  
 tworzenie  
   animacji, 31  
   aplikacji, 330

- App ID, 498
- certyfikatu, 206, 208, 491
- dźwięku, 213
- encji, 275
- identyfikatora typu kuponu, 489
- kontakt, 145
- kuponu, 482
- manifestu, 495
- modelu obiektu zarządzanego, 274
- obiektów zarządzanych, 263
- obiekty obrazka, 388
- obiekty UILabel, 232
- obiekty żądania, 171
- ogłoszenia, 176
- osiągnięć, 122
- paczki kuponu, 489, 495
- programowe kontaktu, 147
- silnika odtwarzania, 152
- układów, 420
- wyzwania, 122
- żądań pobrania, 280
- typ going next, 60
- typy
  - danych, 219, 262, 276
  - dynamiczne, 441

## U

- UIKit Dynamics, 29, 41
- układ przepływu, 415
- umowa o zachowaniu poufności, 17
- uprawnienia, 459
  - do publikowania wiadomości, 334
  - w serwisie Facebook, 331
- uruchamianie
  - kolejki, 365
  - kolejki dyspozytora, 371
  - profilera czasowego, 521
- usługa
  - Core Location, 44
  - iCloud, 181–199
    - inicjalizacja, 184
    - rozwiązywanie konfliktów, 193
    - synchronizacja słowników, 197
    - włączanie obsługi, 184
  - wykrywanie konfliktów, 191
  - wyświetlanie dokumentów, 188

- usługi sieciowe, 475
- ustawienia prywatności, 45
- usuwanie
  - kuponu, 506
  - obiektów zarządzanych, 293
- uwierzytelnianie, 95, 97, 115
- uzyskiwanie położenia użytkownika, 44

## W

- walidator, 260
- warstwa, 56, 84
- warstwy map, 63
- wąskie gardła, bottlenecks, 520
- wątek, 311
- wersjonowanie modeli, 262
- widok
  - adnotacji, 59, 62
  - assetGroupTableView, 461
  - Build Settings, 306
  - debugowania, 511
  - dekoracyjny, 416
  - kolekcji, 407, 413, 424–427
  - kont, 183, 184
  - osiągnięcia, 126
  - Review Changes, 306
  - tableView, 230
  - UITableView, 137, 296
  - UITextView, 249
  - warstw, 64
  - wyświetlenia filmu, 283
  - zasobu, 469
- wielowartości, 139, 148
- wielowartość ABMultiValueRef, 141
- wielozadaniowość, multitasking, 348
- wirtualne granice, 63, 71, 78
- witryna iTunes Connect, 90
- właściwości, properties, 299
  - ABRecordRef, 138
  - elementu, 39
  - klasy UIDynamicItem, 40
- właściwość

- achievementDictionary, 117
  - addressDictionary, 70
  - administrativeArea, 70
  - contentInsert, 250
  - currentPlaybackTime, 155
  - draggable, 62
  - kABPersonAddressProperty, 141
  - printInfo, 249
  - rotation, 449
  - showsCompletionBanner, 120
  - showsPageRange, 249
  - subAdministrativeArea, 70
  - subtitle, 58
  - timeScope, 105
  - title, 58
  - włączanie obsługi iCloud, 184
  - współczynnik obrotu, 449
  - współdzielenie pęku kluczy, 384
  - współrzędne geograficzne, 43
  - wstawianie
    - obiektów, 278
    - obiektu zarządzanego, 293
  - wstrzymywanie gry, 89
  - wybieranie
    - instrumentów, 519
    - kategorii, 396
    - kontaktów, 142
    - losowe, 164
    - partnera, 234
    - utworu, 162
    - znajomych, 280
  - wyciek pamięci, 522
  - wydarzenie, 478
  - wygląd i zachowanie, 130
  - wygląd kuponu, 486
  - wyjątek, 514
  - wykrywanie
    - konfliktów, 191
    - nieudanych gestów, 453
    - partnerów, 237
    - trafień, 434
    - twarży, 403
  - wykrzyknienia, 219
  - wyliczanie zasobów, 458
  - wysyłanie
    - danych, 178, 238, 239
    - komunikatów, 327
    - ogłoszenia, 175
    - powiadomień, 226
    - tweeta, 325
    - wiadomości, 326, 330, 335
    - wyników, 99, 101
  - wyszukiwanie piosenek, 166
  - wyświetlanie
    - danych, 175
    - danych obiektu, 283
    - danych użytkowników, 137
    - dodatkowych informacji, 58
    - dokumentów, 188
    - dymku, 233
    - kaktusa, 83
    - komunikatu, 232
    - kuponu, 481, 505
    - list wyników, 104
    - map, 53
    - nazwy partnera, 242
    - obiektów zarządzanych, 280
    - obrazków, 388
    - trasy, 79
    - warstwy mapy, 64
    - widoków adnotacji, 59, 60
    - zasobów, 466
  - wytyczne interfejsu, 17
  - wyznaczanie tras, 75
  - wyzwania, challenges, 106, 120
  - wznawianie gry, 89
- X**
- Xcode, 261
- Z**
- zabezpieczanie
    - danych, 377
    - słownika, 382
  - zachowania dynamiczne, 30

- zachowanie
  - UIAttachmentBehavior, 35
  - UICollisionBehavior, 33, 34
  - UIGravityBehavior, 32
  - UIPushBehavior, 37
  - UISnapBehavior, 37
- zadania działające w tle, 347
- zakładka
  - Accounts, 183
  - Achievements, 113
  - Capabilities, 499
  - Filmy, 270
- zakres stron, 249
- zapisywanie
  - obrazka, 470
  - zmian, 295
- zarządzanie pamięcią, 138
- zasięg punktu przerwania, 514
- zasoby JSON, 170
- zbiory, 464
- zdalne powiadomienia, 214
- zdarzenie
  - session:peer:didChangeState:, 243
  - UIControlEventTouchDown, 85
- zdjęcia, 457
- zegar, 85, 160
- zgłaszanie osiągnięć, 117
- zliczanie referencji, 302
- zmiana
  - kuponu, 505
  - stanu, 242
  - stanu autoryzacji, 46
  - układu widoku kolekcji, 424
- związek, relationship, 277
- związki pomiędzy obiektami, 276

## Ż

- żądanie, 171, 179
  - pobrania, 265
  - POST, 224



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



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

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

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





## iOS dla ambitnych!

Systemy spod znaku jabłka od zawsze wyznaczały trendy w wyglądzie interfejsu użytkownika. Nie inaczej jest w przypadku urządzeń mobilnych. Platforma iOS zdobyła uznanie użytkowników dzięki niezwykle wygodzie użytkowania, atrakcyjnemu wyglądowi oraz ogromnym możliwościom. Jeżeli chcesz stworzyć aplikację, w której wykorzystasz zaawansowane narzędzia i mechanizmy SDK, to trafisz na doskonałą publikację!

W trakcie lektury będziesz mieć niepowtarzalną okazję dogłębnego poznania systemu iOS – wraz z jego licznymi bibliotekami, ułatwiającymi pracę programistom. Jaką wiedzę zdobędziesz dzięki tej książce? Nauczysz się tworzyć animacje z wykorzystaniem UIKit Dynamics oraz stosować biblioteki Core Location, MapKit i Geofencing. Ponadto dowiesz się, do czego służą Game Center oraz iCloud. W kolejnych rozdziałach znajdziesz informacje poświęcone komunikacji między systemami za pośrednictwem formatu JSON, lokalnym repozytorium danych oraz zaawansowanym operacjom na tekście przy użyciu biblioteki TextKit. Dodatkowo poznasz tajniki SDK i sprytnie techniki pracy z IDE. To obowiązkowa lektura każdego programisty platformy iOS!

Dzięki tej książce:

- poznasz najbardziej przydatne biblioteki platformy iOS,
- wykorzystasz potencjał środowiska Xcode,
- zintegrujesz swoją aplikację z iCloud i Game Center,
- zaznajomisz się z zaawansowanymi funkcjami języka Objective-C,
- stworzysz lepszą aplikację w krótszym czasie.

**helion.pl**  
księgarnia  
internetowa

Nr katalogowy: 24483

Księgarnia internetowa  
<http://helion.pl>

Zamówienia telefoniczne:  
**0 801 339900**  
**0 601 339900**

Informatyka w najlepszym wydaniu



**Helion**

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

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



Addison  
Wesley

cena: 99,00 zł

ISBN 978-83-246-9182-1

