

Wykorzystaj potencjał platformy iOS!



Tworzenie aplikacji na
platformę iOS 5

z wykorzystaniem Xcode, Interface Builder,
Instruments, GDB
oraz innych kluczowych narzędzi

Brandon Alexander • J. Bradford Dillon • Kevin Y. Kim

Apress®



Tytuł oryginału: Pro iOS 5 Tools: Xcode Instruments and Build Tools

Tłumaczenie: Robert Górczyński

ISBN: 978-83-246-4887-0

Original edition copyright © 2011 by Brandon Alexander, J. Bradford Dillon, and Kevin Y. Kim.
All rights reserved.

Polish edition copyright © 2012 by HELION SA.
All rights reserved.

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

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

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

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

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

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/twapxc>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

	O autorach	11
	O recenzencie technicznym	13
	Podziękowania	15
	Wprowadzenie	17
Rozdział 1	Zaczynamy!	19
	Dla kogo jest ta książka?	19
	Czym jest ta książka?	20
	Czego potrzebujesz, aby rozpocząć pracę?	21
	Co znajdziesz w tej książce?	21
	A więc do dzieła!	24
Rozdział 2	Pierwszorzędne narzędzia	25
	Rozejrzyj się	26
	Tak wiele paneli!	29
	Edytory i powiązane z nimi narzędzia	29
	Pasek Jump Bar	31
	Okno Organizer	32
	Karty, karty i jeszcze więcej kart	33
	Wracamy do kodu	34
	Uaktualnienie modelu Core Data	34
	Dodanie nowego kontrolera widoku	39
	Podsumowanie dotychczasowych działań	45
	Praca z instrumentami narzędzia Instruments	45
	Praca z narzędziem Instruments	47
	Tak wiele instrumentów	48
	Dostrajanie wydajności działania	50
	Podsumowanie	51

Rozdział 3	Trzy ekrany i coś... to działa	53
	Praca z GitHub	53
	Nawiązanie połączenia z aplikacją Super Checkout	55
	Rozejrzyj się	58
	Uruchomienie aplikacji Super Checkout	59
	Poruszanie się po projekcie (i po Xcode)	60
	Podsumowanie	61
Rozdział 4	Zarządzanie pamięcią i diagnostyka	63
	Gałęzie to nasi przyjaciele	64
	Automatyczne zarządzanie pamięcią	65
	Konwersja na wersję w technologii ARC	66
	Wykrywanie cykli zachowania	70
	Powrót do ręcznego zarządzania licznikiem użycia obiektu	71
	Najlepsze praktyki w zakresie tworzenia kodu	71
	Przytrzymać czy nie przytrzymać — oto jest pytanie	71
	Łączenie właściwości oraz polimorficzne kropki	73
	Analiza statyczna	76
	Zombie — nie, nie ten rodzaj Zombie	79
	Zombie w innych wątkach	83
	Wycieki	86
	Powracamy do cyklu zachowania	91
	GDB kung-fu	92
	GDB, nie zawieźdź mnie teraz	93
	Rozpoczęcie pracy z GDB	94
	Ustalanie kontekstu — gdzie ja jestem?	95
	Przegląd danych — co otrzymałem?	97
	Wymuszenie awarii nie jest takie trudne	100
	Zebranie wszystkiego w całość	102
	Usunięcie naszego błędu	102
	Kiedy wszystko inne zawodzi	104
	Błąd typu Heisenbug	104
	Telefon do przyjaciela	105
	Rozpoczęcie od nowa	105
	Skończyliśmy... prawie	105
	Podsumowanie	107
Rozdział 5	Core Animation i płynne przewijanie	109
	Wykorzystanie wątku głównego	109
	Poznanie pętli zdarzeń	110
	Optymalizacja wykonywania kodu	111
	Mały skok w bok, czyli wszystko o docelowych rodzajach plików wykonywalnych	113
	Powrót do profilowania	114
	Usprawnienie listy produktów	120

	Co się dzieje w tle podczas przewijania?	123
	Leniwe wczytywanie obrazów	123
	Nigdy więcej niezadowolającego przewijania	133
	Krótki opis technologii Core Graphics	133
	Powrót do narzędzia Instruments	135
	Ucz się od Apple	142
	Podsumowanie	144
Rozdział 6	Sieć, buforowanie i zarządzanie energią	145
	Zrozumienie sieci i buforowania	146
	API po stronie klienta	146
	API po stronie serwera	159
	Spowolnienie sieci za pomocą Network Link Conditioner	163
	Kontrolowanie bufora	164
	Implementacja bufora na dysku	171
	Zarządzanie energią	173
	Sprzęt	173
	Techniki tworzenia kodu	175
	Podsumowanie	180
Rozdział 7	Przygotowanie wersji beta	181
	Zarządzanie testowaniem wersji beta	182
	Zdefiniowanie wersji beta	182
	Znalezienie testerów wersji beta	182
	Przeszkolenie testerów	184
	Tworzenie wersji tymczasowych aplikacji	184
	Certyfikaty, iOS Provisioning Portal, dystrybucja, jejkul!	185
	Alfa, beta, gamma?	203
	Podsumowanie	203
Rozdział 8	Dlaczego to nie działa?	205
	Programowanie techniką Test Driven Development	205
	Kiedy powinienem rozpocząć testowanie?	206
	Xcode ułatwia przeprowadzanie testów	207
	Dopracowanie testów	218
	Wykorzystanie w aplikacji zdobytej wiedzy	220
	Testowanie SCJSONParser	220
	Obiekt atrapa	221
	Testowanie negacji	223
	Negatywne testowanie i użyteczne porażki	224
	Interfejs użytkownika dla testowania i instrument Automation	227
	Rozpoczęcie pracy z instrumentem Automation	229
	Skryptowanie testów interfejsu użytkownika	233
	Wprowadzenie błędu	237
	Potęga automatyzacji	240
	Testuj w taki czy inny sposób	241

Rozdział 9	Czy mogę to jakoś zautomatyzować?	243
	Ciągła integracja	243
	Poznaj aplikację Jenkins	244
	Rozpoczęcie pracy z aplikacją Jenkins	244
	Interfejs aplikacji Jenkins	245
	Egzorcyzmy z demonem aplikacji Jenkins	249
	Utworzenie zadania	251
	Skryptowanie Xcode	257
	Kto zepsuł aplikację?	260
	Kontrola jakości	261
	Łatwiejsza dystrybucja	265
	PackageApplication, dlaczego?	267
	Polecenie xcrun	267
	Utwórz jedynie archiwum	267
	Eksport poza aplikację Jenkins	268
	Czy ktokolwiek może się tym zająć?	270
	Tworzenie conocnych wersji aplikacji	271
	Zabezpieczenie na przyszłość	272
	Co jeszcze możemy zrobić?	273
Rozdział 10	Teraz chcemy wersję dla iPada	275
	Zanim rozpoczniemy tworzenie kodu	275
	Projektowanie dla iPada	276
	Implementacja wersji dla iPada	279
	Modyfikacja docelowego rodzaju pliku wynikowego	279
	Delegat aplikacji i uruchomienie interfejsu użytkownika	280
	Uaktualnienie listy produktów oraz widoku szczegółowych informacji o produkcie	282
	Modyfikacja kontrolera widoku koszyka na zakupy	302
	Ostatnie przemyślenia	304
	Podsumowanie	304
Rozdział 11	Jak mogę się tym podzielić?	307
	Umieszczenie kodu w bibliotece statycznej	308
	Utworzenie biblioteki statycznej	309
	Używanie biblioteki statycznej	315
	Dzielenie się kodem poprzez serwis GitHub	328
	Rejestracja w serwisie GitHub	329
	Tworzenie repozytorium współdzielonego	330
	Wprowadzanie zmian do repozytorium	333
	Używanie funkcji GitHub	334
	Rozsądny wybór licencji	336
	Podsumowanie	336

Rozdział 12	I jeszcze jedno...	337
	Dostosowanie Xcode do własnych potrzeb	337
	Karta Behaviors	338
	Karta Fonts & Colors	338
	Karta Text Editing	340
	Karta Key Bindings	340
	Karta Downloads	342
	Karta Locations	344
	Karta Distributed Builds	345
	Skróty klawiszowe i nie tylko	345
	Migracja z Xcode 3 do Xcode 4	346
	Podaj mi wreszcie te skróty klawiszowe!	347
	Przeglądanie dokumentacji	349
	Podsumowanie	352
	Skorowidz	353

Trzy ekrany i cóż... to działa

W poprzednim rozdziale pokrótce pokazaliśmy Ci, jak używać nowych narzędzi programistycznych przeznaczonych do tworzenia aplikacji na platformę iOS. W tym rozdziale będziemy kontynuować rozpoczęty wcześniej wątek; poznasz kilka doskonałych technik, których zastosowanie w aplikacji spowoduje, że użytkownicy będą często do niej wracali. W kilku kolejnych rozdziałach będziemy zajmowali się aplikacją, która, delikatnie mówiąc, ma parę wad. Niektóre z nich są oczywiste, natomiast inne pozostają trudne do wykrycia. Omawiana przez nas aplikacja będzie wirtualnym sklepem warzywniczym. Wymagania stawiane przed aplikacją to: wyświetlanie informacji o produkcie łącznie z jego obrazem i opisem, możliwość umieszczenia produktu w koszyku na zakupy, przeglądanie koszyka, wreszcie finalizacja zakupu. Na obecnym etapie aplikacja jest w wersji alfa, to znaczy spełnia podstawowe wymagania, ale wymaga jeszcze wiele pracy, przede wszystkim nad poprawieniem wydajności i stabilności.

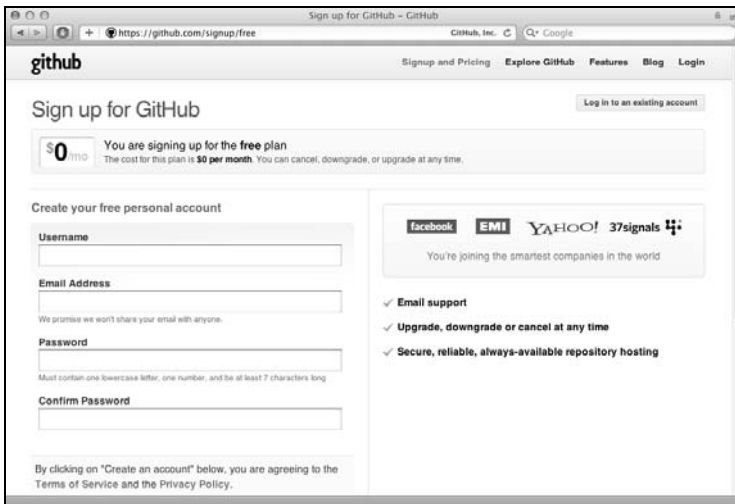
Praca z GitHub

Na potrzeby tego produktu kod źródłowy umieścimy w zewnętrznym repozytorium — wykorzystamy społecznościowy serwis o nazwie GitHub. Bezpłatne konto możesz założyć po przejściu na stronę <https://github.com/signup/free>. W chwili pisania tej książki formularz rejestracyjny wyglądał tak jak na rysunku 3.1.

Serwis GitHub oferuje możliwość założenia bezpłatnego konta pozwalającego na utworzenie nieograniczonej (w chwili powstawania książki) liczby publicznych repozytoriów Git. Oferowane są również płatne konta — w takim przypadku można zaprosić innych programistów do pracy z prywatnymi repozytoriami. Aplikacja, nad którą będziemy pracować w tej książce, zostanie umieszczona w serwisie GitHub jako repozytorium publiczne.

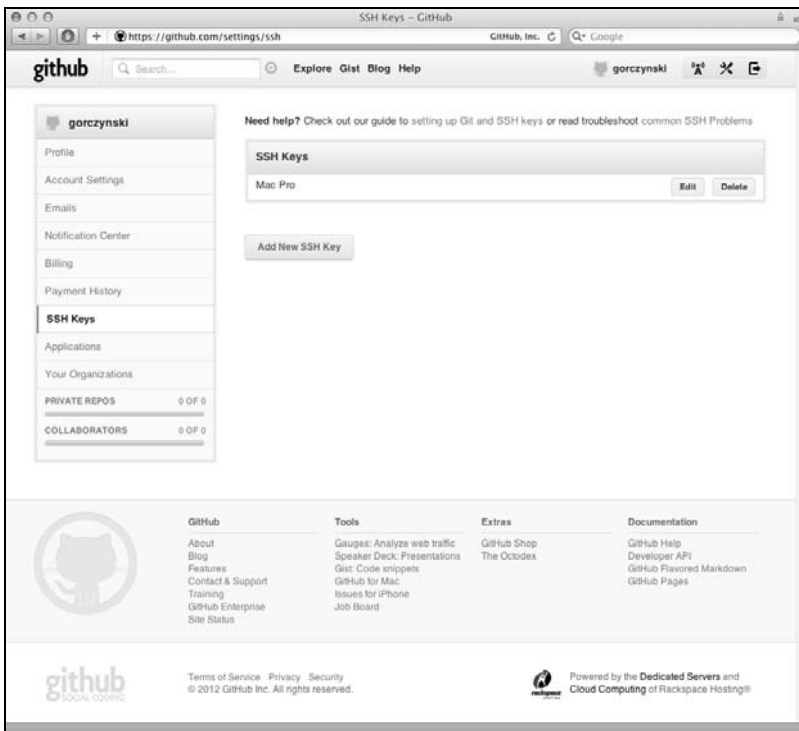
Uwaga!

Git to rozproszony system kontroli wersji. Został wbudowany w narzędzie Xcode, więc aby z niego korzystać, nie musisz niczego instalować. Wbudowaną w Xcode obsługę repozytorium Git będziemy wykorzystywać w książce do śledzenia wprowadzanych zmian. Git oferuje interfejs wiersza poleceń o wspaniałych możliwościach; poznanie choć części z nich na pewno zwiększy Twoją produktywność jako programisty.



Rysunek 3.1. Formularz rejestracyjny konta w serwisie GitHub

Kolejnym krokiem jest przeprowadzenie konfiguracji Twoich publicznych kluczy SSH. Przejdź na stronę <https://github.com/account/ssh>, a następnie kliknij łącze *Add another public key* (zob. rysunek 3.2).



Rysunek 3.2. Dzięki dodaniu klucza publicznego będziesz mógł bezpiecznie pracować z kodem źródłowym poprzez połączenie SSH

Konfiguracja klucza publicznego SSH jest bardzo prostym zadaniem. Cała procedura została przedstawiona na stronie <http://help.github.com/mac-set-up-git/> (zob. rysunek 3.3).



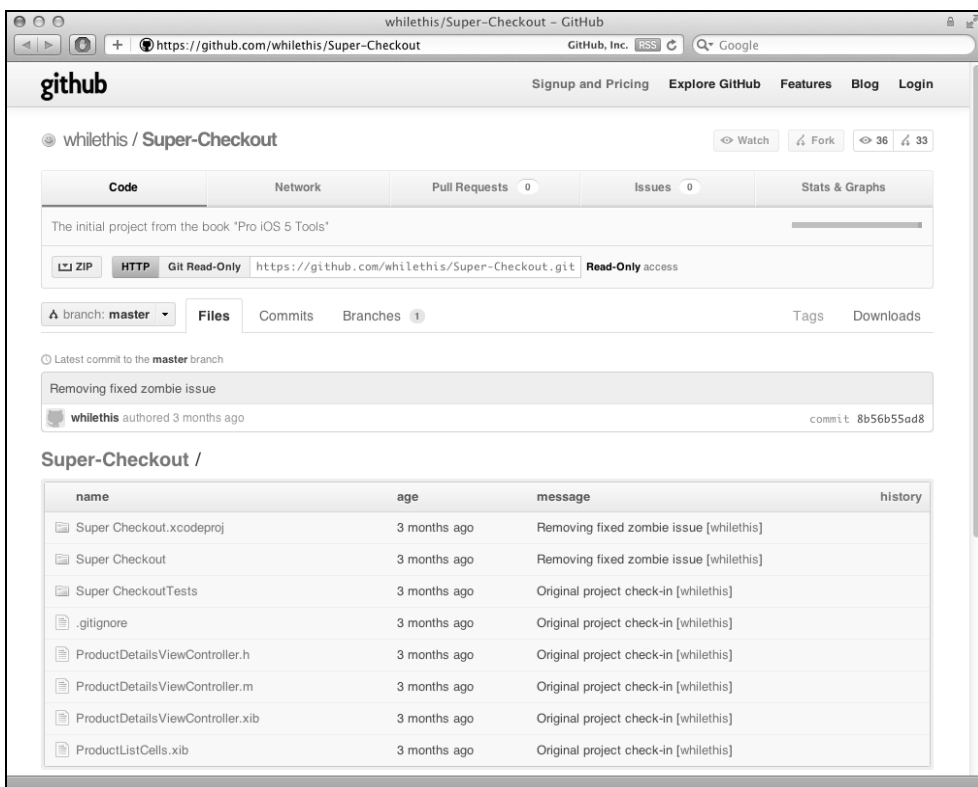
Rysunek 3.3. Konfiguracja kluczy SSH w systemie Mac OS X

Po zakończeniu konfiguracji repozytorium Git oraz kluczy SSH w komputerze można przystąpić do umieszczenia projektu w repozytorium oraz pracy nad projektem z poziomu Xcode.

Nawiązanie połączenia z aplikacją Super Checkout

Kolejnym krokiem jest utworzenie gałęzi projektu, aby móc go pobrać i zacząć wprowadzać własne zmiany. Główny projekt jest dostępny pod adresem <https://github.com/whilethis/Super-Checkout>. Dzięki gałęzi projektu otrzymujesz własną kopię repozytorium, w którym możesz wprowadzać zmiany. Utworzenie gałęzi następuje po kliknięciu przycisku *Fork* znajdującego się w prawym górnym rogu ekranu na stronie głównej projektu (zob. rysunek 3.4).

Na stronie utworzonej gałęzi projektu skopiuj adres URL, który będzie podobny do pokazanego na rysunku 3.5.



Rysunek 3.4. Kliknięcie przycisku Fork na stronie głównej projektu w GitHub powoduje utworzenie własnej kopii repozytorium

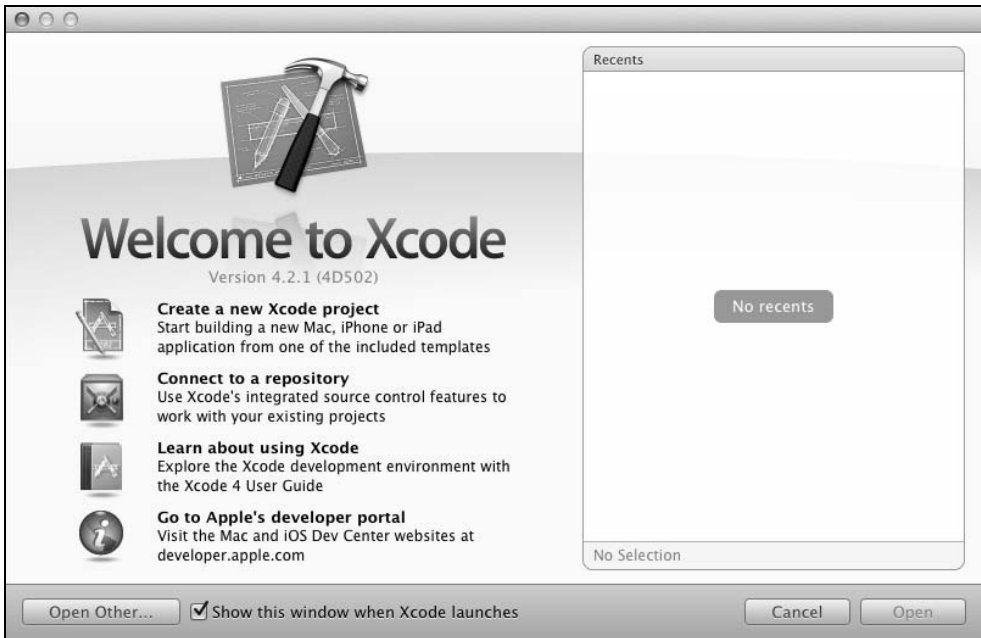


Rysunek 3.5. W adresie URL znajdzie się Twoja nazwa użytkownika serwisu GitHub zamiast pokazanej na rysunku nazwy gorczynski

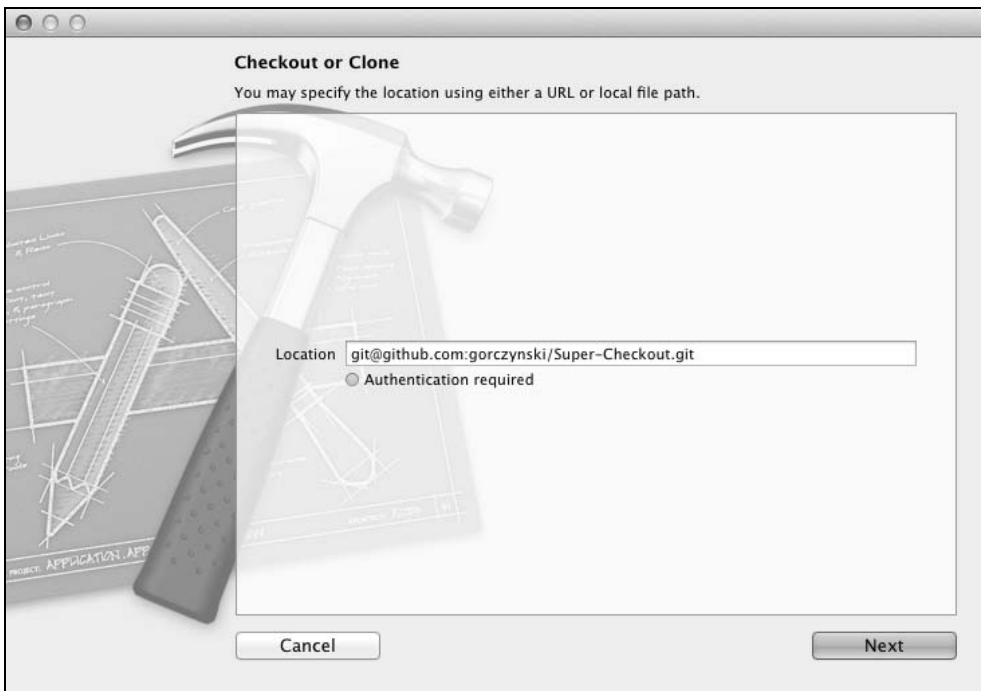
Teraz uruchom narzędzie Xcode 4. Na ekranie wyświetli się pokazany na rysunku 3.6 ekran powitalny. W poprzednim rozdziale utworzyliśmy nowy projekt od podstaw. Tym razem nawiążemy połączenie z repozytorium.

Kliknij przycisk *Connect to a repository*. Zostaniesz poproszony o podanie położenia repozytorium (zob. rysunek 3.7). W tym miejscu wklej adres URL skopiowany z serwisu GitHub; Xcode automatycznie sprawdzi dostępność projektu i pozwoli kontynuować pracę. Kliknięcie przycisku *Next* spowoduje zapis kodu na dysku.

Jeżeli z jakiegokolwiek powodu spróbujesz nawiązać połączenie z serwisem GitHub przed przeprowadzeniem konfiguracji kluczy SSH, zostaniesz poproszony o potwierdzenie tożsamości w tym serwisie poprzez podanie nazwy użytkownika i hasła. Celem przeprowadzenia wspomnianej wcześniej konfiguracji kluczy SSH jest uniknięcie konieczności podawania danych uwierzytelniających. Jeśli jednak skonfigurowałeś klucze SSH i nie wydałeś polecenia `ssh -T`, to w pokazanym na rysunku 3.7 oknie dialogowym pojawi się informacja, że musisz potwierdzić tożsamość.



Rysunek 3.6. Ekran powitalny Xcode pozwala na utworzenie nowego projektu bądź nawiązanie połączenia z repozytorium

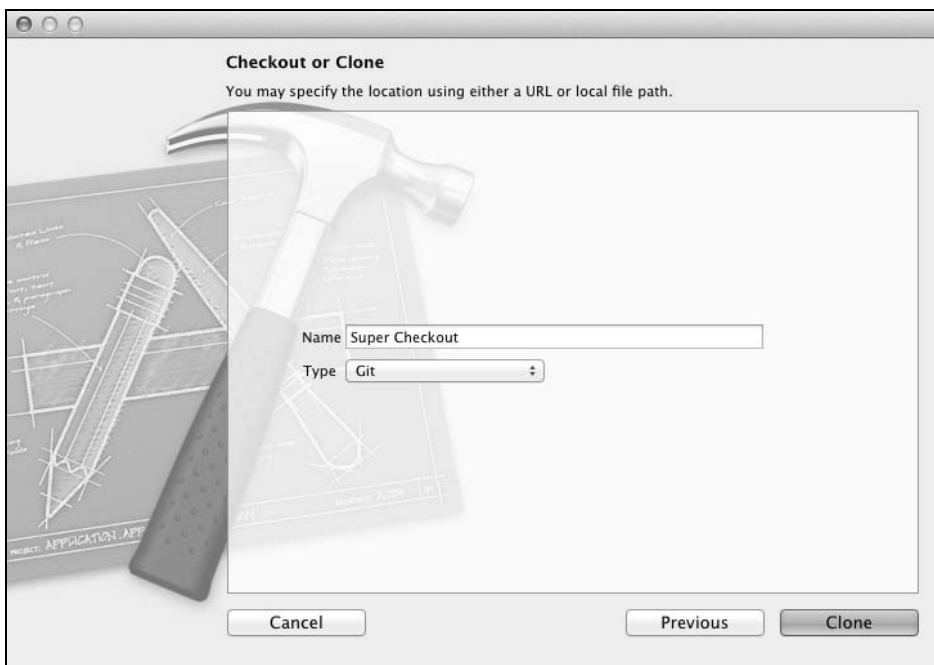


Rysunek 3.7. Xcode automatycznie wykrywa typ repozytorium podczas pobierania kodu

Uwaga!

Podczas bezpośredniego łączenia się ze zdalnym repozytorium upewnij się, że wybrałeś odpowiedni typ repozytorium. Xcode spróbuje klonować pliki lub przekazać je do repozytorium, używając wskazanego protokołu (zob. rysunek 3.8).

Kolejnym krokiem jest nadanie projektowi nazwy w Xcode oraz wskazanie typu repozytorium (zob. rysunek 3.8). Wpisz nazwę *Super Checkout* i kliknij przycisk *Clone*.

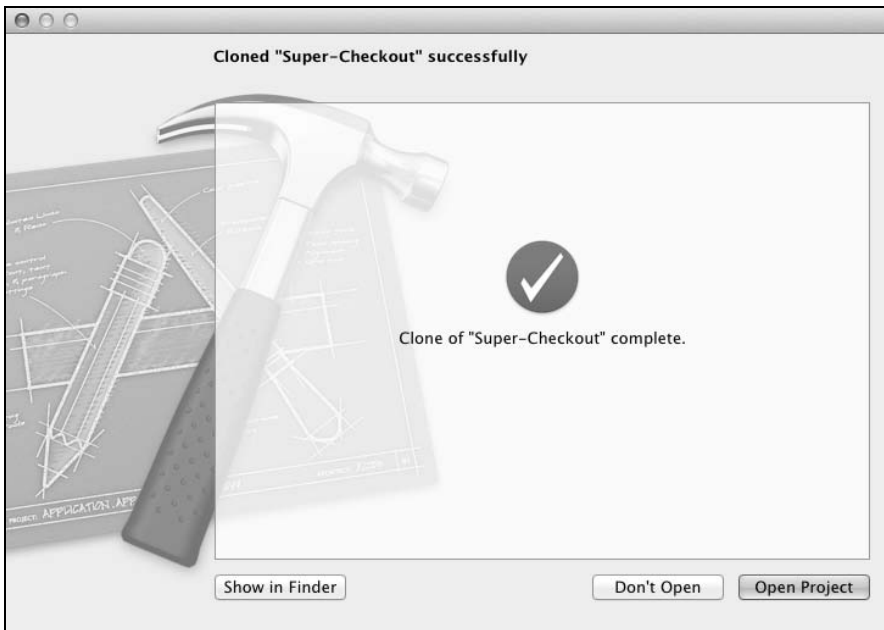


Rysunek 3.8. Nadanie projektowi nazwy i wybór typu repozytorium

W następnym oknie dialogowym musisz podać położenie, w którym zostanie zapisana Twoja kopia repozytorium. Wybierz katalog na dysku i kliknij przycisk *Clone*. Po zakończonej powodzeniem operacji klonowania zobaczysz okno dialogowe z przyciskami *Open Project* i *Don't Open* (zob. rysunek 3.9). Kliknij przycisk *Open Project*; teraz jesteś gotowy do rozpoczęcia pracy nad projektem. Jeśli zrezygnowałeś z omówionego wcześniej kroku utworzenia kluczy publicznych, zobaczysz pewne pola dialogowe związane z podaniem i potwierdzeniem tożsamości. Gdy napotkasz jakiegokolwiek problemy, wróć do kroku tworzenia kluczy publicznych i spróbuj jeszcze raz.

Rozejrzyj się

Aplikacja jest bardzo prosta. Składa się z kontrolera nawigacyjnego, kontrolera widoku tabeli odpowiedzialnego za wyświetlanie dostępnych produktów, kontrolera widoku szczegółowego wyświetlającego informacje o wybranym produkcie oraz kontrolera koszyka na zakupy. Ostatni z wymienionych jest kontrolerem widoku modalnego. Moduł odpowiedzialny za kontakt z usługą jest prostym silnikiem i używa tego samego podejścia, które Matt Gemmell (<http://instinctivecode.com/>) zastosował w opracowanym przez siebie silniku aplikacji obsługującej serwis Twitter.



Rysunek 3.9. Klonowanie projektu zakończyło się powodzeniem

API silnika zapewnia komunikację z serwerem i wywołuje analizator w celu przetworzenia danych pochodzących z serwera. Serwer przekazuje dane w formacie JSON (ang. *JavaScript Object Notation*), natomiast projekt wykorzystuje bibliotekę SBJSON (<https://github.com/stig/json-framework/>) do przetwarzania otrzymanych danych w rodzimych klasach Cocoa Touch. Każdy kontroler widoku otrzymuje własny egzemplarz silnika oraz stosuje mechanizm delegacji w celu zapewnienia asynchronicznej komunikacji z serwerem.

Interfejs użytkownika to przede wszystkim widok tabeli wraz z kilkoma własnymi komórkami zaprojektowanymi w module Interface Builder.

Uruchomienie aplikacji Super Checkout

Uruchom aplikację w symulatorze lub urządzeniu iOS — powinna wyglądać tak jak na rysunku 3.10. Pierwszym spostrzeżeniem jest opóźnienie we wczytywaniu obrazów (ten problem rozwiążemy w jednym z kolejnych rozdziałów). Naciśnięcie obrazka produktu powoduje wyświetlenie szczegółowych informacji o produkcie, natomiast naciśnięcie przycisku *Cart* umieszcza dany produkt w koszyku. Warto w tym miejscu dodać, że naciśnięcie przycisku *Cart* powoduje „obrót” widoku i wyświetlenie zawartości koszyka na zakupy.

Kiedy zaczniesz poruszać się po aplikacji i poznasz ją lepiej, zobaczysz, że szybki powrót z ekranu zawierającego informacje szczegółowe o produkcie do listy produktów najczęściej powoduje awarię aplikacji. Szczegrze mówiąc, aplikacja ulega awarii także w innych przypadkach — to naprawdę wersja alfa, która wymaga jeszcze sporo pracy.

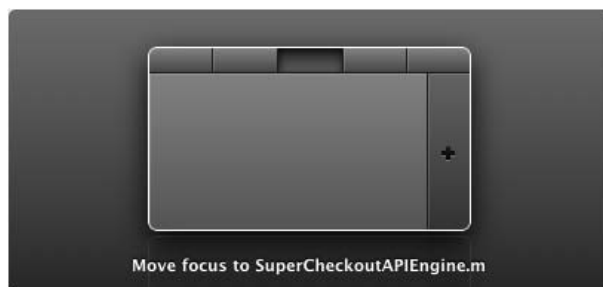


Rysunek 3.10. Aplikacja Super Checkout w wersji alfa

Poruszanie się po projekcie (i po Xcode)

Zarządzanie pracą w Xcode jest kluczem do osiągnięcia biegłości w tworzeniu aplikacji i usprawnieniu każdego projektu. Umiejętność przemieszczania się między poszczególnymi edytorami bez konieczności odrywania rąk od klawiatury może znacznie zwiększyć szybkość prac nad projektem, a poza tym jest to po prostu wygodne. W Xcode 4 wprowadzono wiele użytecznych funkcji pozwalających na pełne wykorzystanie nowego interfejsu. W stosunku do Xcode 3 zmianie uległa także spora liczba najczęściej używanych skrótów klawiszowych.

Jednym z najużyteczniejszych skrótów klawiszowych w Xcode 4 jest prawdopodobnie *Command+J*. Po jego naciśnięciu wyświetla się okno *Move focus to...* (zob. rysunek 3.11). Podświetlony obszar wskazuje miejsce, w którym się znajdziesz po naciśnięciu klawisza *Enter*. Poruszanie się między kartami jest bardzo proste i odbywa się za pomocą klawiszy kursora. Przejście do prawej krawędzi bieżącego edytora (obszar ze znakiem plusa) powoduje utworzenie nowego edytora asystenta.



Rysunek 3.11. Funkcja Move focus to... pozwala na poruszanie się między kartami i edytorami asystentami

Przejdźcie między plikiem implementacji i plikiem nagłówkowym następowo po naciśnięciu klawiszy *Command+Option*¹+kursor w górę. W Xcode 4 ten skrót został zamieniony na *Command+Control*+kursor w górę. Można również użyć gestu machnięcia trzema palcami po gładziku w górę bądź w dół, jeśli ten gest nie został już przypisany innej funkcji w systemie operacyjnym (system Mac OS X 10.7 Lion intensywnie wykorzystuje gest machnięcia trzema palcami).

Kliknięcie symbolu z jednocześnie naciśniętym klawiszem *Option* powoduje wyświetlenie małego okna dialogowego zawierającego informacje na temat danego symbolu. Z poziomu tego okna można wyświetlić plik nagłówkowy klasy bądź przejść do dokumentacji, pod warunkiem że jest to klasa należąca do struktury dostarczanej przez Apple.

Kliknięcie symbolu z jednocześnie naciśniętym klawiszem *Command* powoduje przejście do deklaracji tego symbolu. Jeżeli symbol jest zmienną, podświetlona zostanie deklaracja tej zmiennej. Jeśli symbol jest klasą, w edytorze wyświetli się plik nagłówkowy, w którym został zdefiniowany dany symbol.

Kliknięcie symbolu z jednocześnie naciśniętymi klawiszami *Option* i *Command* powoduje wyświetlenie edytora asystenta i przeprowadzenie takiej samej operacji, jak w przypadku kliknięcia z naciśniętym jedynie klawiszem *Command*.

Naciśnięcie klawiszy *Command+Shift+O* powoduje wyświetlenie okna dialogowego *Open Quickly*. W nim możesz zacząć wprowadzać początek nazwy pliku, dopasowane nazwy plików zostaną wyświetlone w dolnej części okna. To bardzo wygodny sposób, pozwalający na szybkie otwieranie plików.

Opisane powyżej skróty klawiszowe (oraz wiele innych) przedstawiono w rozdziale 12.

Podsumowanie

Aplikacja przedstawiona w tym rozdziale znajduje się na etapie alfa i wymaga usprawnienia na wielu różnych obszarach. Naszym celem jest przygotowanie produktu końcowego, czyli aplikacji, która nie będzie ulegała awarii i będzie miała poprawną architekturę. W kolejnych rozdziałach przeprowadzimy więc operacje m.in.: profilowania, refaktoringu oraz przepisania na nowo pewnych fragmentów aplikacji. Odbywamy dość długą podróż, więc przewróć kartkę i zacznij pracę!

¹ W nowszych klawiaturach Apple klawisz *Option* jest opisany jako *Alt* — *przyj. tłum.*

Skorowidz

__autoreleasing, 66
__strong, 65
__unsafe_unretained, 66
__weak, 66
2G, 173, 174
3G, 173

A

abort(), 36, 37
analyzer statyczny, 76, 77, 78, 79
ARC, 63, 65, 66
 konwersja aplikacji, 66, 67, 68
asercja, 213
ASIDownloadCache, klasa, 171
ASIHTTPRequest, 149, 171
automatyczne zarządzanie pamięcią, *Patrz* ARC
autorelease, polecenie, 72
autorelease, pula, 72, 73

B

beta, wersja, 181
 dystrybucja tymczasowa, 197, 198, 199
 identyfikator aplikacji, 196, 197
 identyfikatory UDID, 192, 193
 przeszkolenie testerów, 184
 rozprowadzenie, 201, 202
 tworzenie, 185, 200
 tworzenie certyfikatu, 188, 190, 191, 192
 wybór testerów, 182, 183
 wymagania sprzętowe, 182
 zarządzanie testowaniem, 182
 zdefiniowanie, 182
 żądanie certyfikatu, 185, 186, 187

biblioteka statyczna, 308
 pliki nagłówkowe, 312
 tworzenie, 309, 310
 używanie, 315, 316
biblioteki sieciowe, 159
buforowanie, 164, 165
BUILD_PRODUCTS_DIR, 328

C

c, polecenie, 98
Charles, aplikacja, 162
CI, *Patrz* ciągła integracja
ciąg Fibonacciego, 117
ciągła integracja, 243, 244
Clang Scan-Build, 261, 262
Clang Static Analyzer, 261
Continuous Integration, *Patrz* ciągła integracja
Core Animation, 134
Core Graphics, 133, 134
cykl zachowania, 70, 71, 91, 92
cykl życiowy aplikacji, 111

D

dokumentacja, przeglądanie, 349, 351
drawRect, metoda, 133

E

edytory, Xcode 4, 29, 30
energia, zarządzanie, 173, 174, 179
 2G, 173, 174
 3G, 173
 praca w sieci, 175

energia, zarządzanie
 śledzenie zużycia, 176, 177, 179
 Wi-Fi, 174

F

Fibonacciego, ciąg, 117
Fix-it, funkcja, 25

G

GDB, 93, 94, 102
 c, polecenie, 98
 info args, polecenie, 97
 info locals, polecenie, 97
 n, polecenie, 103
 po, polecenie, 98
 whatis, polecenie, 97
Git, 53
 przekazywanie plików, 38
Git Plugin, 249, 253
GitHub, 53, 54, 328, 329
 rejestracja w serwisie, 329
 śledzenie problemów, 336
 tagi, 334, 335
 tworzenie repozytorium, 330
 wybór licencji, 336
 zmiany w repozytorium, 333
GNU Debugger, *Patrz* GDB

H

Heisenbug, błąd, 104
Hudson, 244

I

IDE, 17
info args, polecenie, 97
info locals, polecenie, 97
instruments, polecenie, 273
Instruments, Xcode 4, 25, 45, 47, 48
 Activity Monitor, 49, 175
 Allocations, 49
 Automation, 49, 227, 228, 229, 230, 231
 Core Animation, 50, 135, 136

Energy Diagnostics, 49, 176
Leaks, 47, 49
OpenGL ES Analysis, 50
OpenGL ES Driver, 50
System Usage, 49
Time Profiler, 49, 117, 118
Interface Builder, 25
iOS Developer Program, 21
iOS Provisioning Portal, 181
iPad
 implementacja projektu, 279
 projektowanie interfejsu, 276, 278

J

Jenkins, 244
 About Jenkins, 247
 Build Details, 255
 Clang Scan-Build, 261, 262
 Clang Static Analyzer, 262
 Configure System, 245
 definiowanie zadania, 252
 Git Plugin, 249, 253
 instalacja, 245
 interfejs, 245
 Jenkins CLI, 246
 Job Configuration, 252
 Job Details, 254
 Load Statistics, 246
 Manage Nodes, 246
 Manage Plugins, 246
 minimalne wymagania, 244
 Poll SCM, 256
 Prepare for Shutdown, 247
 Reload Configuration from Disk, 246
 rodzaje zdarzeń, 251
 Script Console, 246
 System Information, 246
 System Log, 246
 tworzenie nowego użytkownika, 244, 245
 wtyczki, instalacja, 248, 249
 zadanie, 251
 zmiennne środowiskowe, dodawanie, 247, 248
Jump Bar, pasek, 25, 31, 32

K

Knuth, Donald, 111
 kompilacja, 34
 schematy, 46, 113
 kompilator LLVM 3.0, 25
 kompozycja widoku, 137
 komunikacja sieciowa, 146
 API po stronie klienta, 146, 147
 API po stronie serwera, 159
 spowolnienie sieci, 163
 kropka polimorficzna, 73

L

launchctl, narzędzie, 249
 layer, właściwość, 134
 layoutSubviews, metoda, 133, 134
 licznik użycia obiektu, 71
 LLVM 3.0, kompilator, 25
 logFail(), 238
 logStart(), 235

N

n, polecenie, 103
 NARC, 71
 Network Link Conditioner, 163
 NSAssert, 213, 216
 NSCache, 165

O

obiekt atrapa, 221
 obrazy, leniwe wczytywanie, 123
 OCUnt, 207, 213
 okno preferencji, Xcode 4, 337
 Behaviours, karta, 338
 Distributed Builds, karta, 345
 Downloads, karta, 342
 Fonts & Colors, karta, 338, 339
 Key Bindings, karta, 340
 Locations, karta, 344
 Text Editing, karta, 340
 Organizer, Xcode 4, 32
 Devices, karta, 32
 Documentation, karta, 33

Project, karta, 32
 Repositories, karta, 32
 outlety właściwości, 40

P

PackageApplication, polecenie, 266, 267
 pamięć
 automatyczne zarządzanie, *Patrz* ARC
 ręczne zarządzanie, 71
 wycieki, 86, 88, 89, 90, 91
 zwalnianie, 71, 72
 panel nawigacyjny, Xcode 4, 29
 Breakpoint, karta, 29
 Debug, karta, 29
 Issue, karta, 29
 Log, karta, 29
 Project, karta, 29
 Search, karta, 29
 Symbol, karta, 29
 pasek narzędziowy, Xcode 4, 29
 pętla działania, *Patrz* wątek główny
 pętla zdarzeń, *Patrz* wątek główny
 po, polecenie, 98
 powiadomienia, 260
 przestrzeń robocza, 324

Q

Quartz 2D, 134

R

Release, tryb, 111, 112
 RESTClient, 161
 Retain Cycle, *Patrz* cykl zachowania
 ręczne zarządzanie pamięcią, 71

S

scan-build, polecenie, 261, 272
 schematy kompilacji, 113
 SenTestCase, 213
 SenTestingKit, 213
 setNeedsDisplay, metoda, 133
 setNeedsLayout, metoda, 133, 134

sieć, obsługa, 146

API po stronie klienta, 146, 147

API po stronie serwera, 159

biblioteki, 159

spowolnienie sieci, 163

skrótów klawiszowe, 60, 347, 348, 349

STAssert, 216

STAssertEquals, 214

STAssertNoThrow, 214

STAssertNotNil, 213

STAssertTrue, 214

STFail, 214

Stocks, aplikacja, 142, 143, 144

Subversion, 64

system śledzenia błędów, 184

systemy kontroli wersji, 64

T

TDD, *Patrz* Test Driven Development

Test Driven Development, 205, 206, 207

testerzy, 181

przeszkolenie, 184

wybór, 182, 183

TestFlight, 270, 271

testowanie, 206, 207, 241

interfejsu użytkownika, 207

testy jednostkowe, 207, 227

dołączanie do projektu, 207, 208, 209

konfiguracja, 210

Torvalds, Linus, 308, 329

U

UIAApplication, 228

UIAccessibility, 227

UIAElement, 227

UIAHost, 228

UIALogger, 228

UIATarget, 228

UIAutomation, 227, 228

UILabel, 137

UINavigationController, 134

UIScrollView, 134

UITableView, 134

UIView, 133, 137

W

wątek główny, 109, 110, 111

wersja beta, 181

dystrybucja tymczasowa, 197, 198, 199

identyfikator aplikacji, 196, 197

identyfikatory UDID, 192, 193

przeszkolenie testerów, 184

rozprowadzenie, 201, 202

tworzenie, 185, 200

tworzenie certyfikatu, 188, 190, 191, 192

wybór testerów, 182, 183

wymagania sprzętowe, 182

zarządzanie testowaniem, 182

zdefiniowanie, 182

żądanie certyfikatu, 185, 186, 187

whatis, polecenie, 97

właściwości, 73, 75

wycieki pamięci, 86, 88, 89, 90, 91

wydajność, dostrajanie, 50

X

Xcode 4, 21

__autoreleasing, 66

__strong, 65

__unsafe_unretained, 66

__weak, 66

analizator statyczny, 77, 78, 79

ARC, 63, 65, 66

asystent, 25

BuildSettings, karta, 113

Core Data, modyfikacja modelu, 34

dokumentacja, przeglądanie, 349, 351

edytory, 29, 30

Fix-it, funkcja, 25

gałęzie, łączenie, 105, 106, 107

gałęzie, tworzenie, 64, 65

Git, przekazywanie plików, 38

Instruments, 25, 45, 47, 48, 49

Interface Builder, 25

Jump Bar, pasek, 25, 31, 32

karty, 33

kategorie, tworzenie, 169

kompilacja, 34

kontroler widoku, dodawanie, 39

LLVM 3, kompilator, 25
migracja z Xcode 3, 346
nowe funkcje, 25
nowy projekt, 26, 27
okno główne, 27, 28
okno preferencji, 337
Organizer, okno, 32
panel nawigacyjny, 29
pasek narzędziowy, 29
połączenie z repozytorium, 56, 58
przestrzeń robocza, 324
schematy kompilacji, 46
Show Quick Help, przycisk, 31
skrótów klawiszowe, 60, 61, 345, 346, 347,
348, 349

xcodebuild, 257, 258, 259
xcodebuild, 257, 258, 259
opcje, 257, 258
xcode-select, narzędzie, 272
xcrun, polecenie, 267

Z

zarządzanie pamięcią, 71, 73
właściwości, 73, 75
Zombie, wykrywanie, 79, 80, 81, 82, 83

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>

System operacyjny iOS, wykorzystywany w urządzeniach Apple, cały czas zyskuje na popularności. Jego udział w rynku jest ogromny, a z każdym dniem staje się coraz większy. Co sprawia, że użytkownicy go uwielbiają? Atrakcyjna szata graficzna, błyskawiczne reagowanie na polecenia użytkownika oraz ogromny wybór dopracowanych aplikacji to główne atuty decydujące o jego sukcesie.

Aby zagwarantować ciągle dopływ aplikacji wysokiej jakości, twórcy tego systemu udostępniłi programistom świetne środowisko do ich wytwarzania. W trakcie lektury tej książki zaznajomisz się z możliwościami systemu. Dowiesz się, jak używać wszystkich narzędzi dostępnych w arsenale programisty iOS: Xcode, Interface Builder, Instruments, a także narzędzi służących do diagnostyki sieci, obsługi systemu kontroli wersji i innych. Poznasz techniki usuwania błędów z aplikacji, znajdziesz informacje o strukturach Cocoa, zarządzaniu pamięcią, technologii ARC oraz automatyzacji. Nauczysz się tworzyć atrakcyjny i wydajny interfejs użytkownika, rozwiązywać typowe problemy oraz szanować baterię użytkownika. Na szczególną uwagę zasługuje rozdział poświęcony testowaniu aplikacji oraz najlepszym technikom rozprowadzania jej wśród beta-testerów. Książka ta jest idealną pozycją dla każdego programisty chcącego stworzyć jedyną w swoim rodzaju aplikację dla platformy iOS 5.

Odkryj tajemnice iOS 5:

- **Zaprojektuj atrakcyjny i wydajny interfejs użytkownika**
- **Zapewnij niezawodność dzięki korzystaniu z testów**
- **Rozwiąż typowe problemy związane z wyciekami pamięci**
- **Dostarcz aplikację beta-testerom**

Zaskocz użytkowników nowatorską aplikacją!

helion.pl
księgarnia
internetowa

Nr katalogowy: 11680



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/novosci>

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

sięgnij po **WIĘCEJ**



KOD KORZYŚCI

ISBN 978-83-246-4887-0



Cena: 69,00 zł

9 788324 648870

Informatyka w najlepszym wydaniu