ROZDZIAŁ 1

Wprowadzenie

Początki tej książki były dość niewinne, gdyż w marcu 2011 roku Brian wysłał do Toma wiadomość e-mail. Brian pomyślał, że dobrym pomysłem byłoby dodanie kilku linijek do drugiego wydania innej książki o komunikacji bliskiego zasięgu, *Making Things Talk*, nad którą pracowaliśmy w tamtym czasie. W książce był już rozdział poświęcony identyfikacji radiowej (RFID), więc jak trudne mogłoby to być? Dwa i pół roku później nauczyliśmy się wiele o NFC i zdobyliśmy doskonałego i kompetentnego współpracownika, Dona Colemana, autora wtyczki NFC dla PhoneGap.

Chociaż NFC ma duży potencjał, większość materiałów napisanych do tej pory na jego temat nie została napisana dla zwykłego programisty. Wszystko tam zakładało, że jeśli chcesz wiedzieć o NFC, jesteś przygotowany do zrobienia tego od podstaw. Trzeba było zrozumieć szczegóły różnych specyfikacji RFID i być przygotowanym na pisanie kodu, który interpretował strumień bajtów z czytnika NFC po jednym bajcie na raz. Chociaż warto to zrozumieć, doszliśmy do wniosku, że NFC znalazłoby szersze zastosowanie, gdyby programiści mogli skoncentrować się na tym, do czego go używają, a nie na szczegółach niskiego poziomu. Biblioteka PhoneGap Dona była najlepszym narzędziem, jakie znaleźliśmy. Pozwala ona projektować wymiany NFC w sposób, który wyobrażamy sobie jako zamierzony przez projektantów forum NFC: myślisz o wymienianych wiadomościach i nie martwisz się o resztę.

Większość tej książki jest napisana w tym duchu. Poznasz podstawy formatu wymiany danych NFC (NDEF), odczytując i zapisując wiadomości z urządzenia do tagu i z urządzenia do urządzenia. Zobaczysz kilka przykładowych aplikacji - niektóre napisane dla PhoneGap, niektóre dla Arduino, a niektóre dla Node.js - działających na urządzeniach wbudowanych, takich jak Rasp-Berry Pi i BeagleBone Black. Poznasz niektóre wzorce użycia NDEF i dowiesz się, jak możesz myśleć o fizycznej interakcji aplikacji opartych na NFC.

Stan techniki różni się jednak w zależności od platformy. Nie wszystko, co opisują specyfikacje NFC Forum, jest jeszcze dostępne dla zwykłego programisty na każdej

platformie. Staraliśmy się przedstawić mapę drogową w tej książce, szczególnie w sekcji

w późniejszych rozdziałach, jaki jest obecny stan rozwoju i gdzie jest jeszcze miejsce na poprawę użyteczności.

Mamy nadzieję, że ta książka pomoże zwykłym programistom zorientować się, co można zrobić za pomocą NFC, i że zainspiruje bardziej profesjonalnych programistów do tworzenia prostych w użyciu narzędzi, które pomogą rozpowszechnić jego użycie.

Dla kogo jest ta książka

Nie musisz być wyszkolonym profesjonalnym programistą, aby przeczytać tę książkę. Staraliśmy się napisać ją dla entuzjastów programowania - ludzi, którzy zdobyli trochę wiedzy po drodze, ale być może nie w formalnym środowisku nauczania. Nie nauczysz się tutaj pisać kodu na poziomie korporacyjnym, ale otrzymasz praktyczne wprowadzenie do tego, czym jest komunikacja bliskiego zasięgu i jak programować aplikacje wykorzystujące ją na Androidzie, Arduino i wbudowanym Linuksie.

Zakładamy jednak, że masz pewne obycie z programowaniem. Będziesz chciał znać JavaScript i HTML dla większości przykładów w książce. W projektach Arduino poznasz trochę języka C, ale jeśli znasz JavaScript lub Javę, będzie to wyglądać wystarczająco znajomo. W przypadku tych ostatnich projektów powinieneś być trochę zaznajomiony z elektroniką, ale nie musisz.

Zalecana lektura

"Co? Muszę przeczytać inne książki, aby przeczytać tę książkę?" Nie, ale jest kilka książek, które okazały się pomocne przy pisaniu tej książki. Pomyśleliśmy, że mogą one również okazać się przydatne.

Jeśli jesteś nowy w JavaScript, przeczytaj Douglasa Crockforda *JavaScript: The Good Parts*. Pomyśl o tym, przeczytaj to, jeśli jesteś starą ręką w JavaScript. Sprawi, że będziesz lepszym programistą. Wyjaśnia teoretyczne podstawy języka i najlepsze wzorce użycia w sposób jasny i ostateczny.

W przypadku PhoneGap i Androida najbardziej aktualne są internetowe przewodniki dla początkujących; zobacz portal dla programistów PhoneGap i witrynę dla programistów Androida. Aby uzyskać bardziej szczegółowe wprowadzenie do Androida, zobacz *Professional Android 4 Application Develop- ment* lub *Android Programming: The Big Nerd Ranch Guide*.

Aby uzyskać dogłębne wprowadzenie do NFC z perspektywy inżynierskiej, specyfikacje NFC Forum są oryginalnym materiałem źródłowym (niektóre z nich powieliliśmy w Załączniku A dla ułatwienia). Znaleźliśmy również *Professional NFC Application Devel- opment* autorstwa Vedat Coskun, Kerem Ok i Busra Ozdenizci jako dobre odniesienie, szczególnie dla doświadczonych programistów Java. Celowo przyjęliśmy bardziej populistyczne podejście niż ta książka, ponieważ wielu naszych czytelników to hobbyści, hakerzy i inni samozwańczy programiści-dyletanci.

Jeśli dopiero zaczynasz przygodę z Arduino, książka Massimo Banziego *Getting Started with Arduino* jest doskonałym punktem wyjścia. Tom Igoe's *Making Things Talk, 2nd Edition* to dobra książka dla doświadczonych programistów, aby dowiedzieć się o podłączaniu projektów Arduino do sieci. *Książka kucharska Arduino* Michaela Margolisa zawiera również kilka przydatnych przepisów na programy Arduino.

Wprowadzenie do Node.js, które pojawia się w dalszej części tej książki, Brett McLaughlin's *What is Node?* to przyjemne wprowadzenie o długości eseju bez kodu. *The Node Beginner Book* Manuela Kiesslinga i *Hands-On Node.js* Pedro Teixeiry są pomocnymi i krótkimi przewodnikami po rozpoczęciu pracy z rzeczywistym kodem.

Aby uzyskać dobre wprowadzenie do Raspberry Pi lub BeagleBone Black, z którymi spotkasz się w rozdziale 9, możesz znaleźć materiały na początek w samouczkach Adafruit. Dobrym wprowadzeniem są również książki *Getting Started with Raspberry Pi* autorstwa Matta Richardsona i Shawna Wallace'a oraz *Getting Started with BeagleBone* autorstwa Matta Richardsona.

Co zostało omówione w tej książce

Rozdział 2 zawiera wprowadzenie do komunikacji bliskiego zasięgu (NFC) poprzez porównanie jej do identyfikacji radiowej (RFID). Mówiąc najprościej, NFC jest supersetem RFID. Może zrobić większość rzeczy, które może zrobić RFID krótkiego zasięgu, a nawet więcej. Zapoznasz się z najważniejszymi terminami, przyjrzysz się architekturze systemu NFC i dowiesz się, jakich narzędzi potrzebujesz i gdzie je zdobyć.

Rozdział 3 przedstawia PhoneGap i wtyczkę NFC dla PhoneGap. Zainstalujesz narzędzia niezbędne do tworzenia aplikacji PhoneGap na Androida oraz zbudujesz i uruchomisz kilka pierwszych aplikacji. Pod koniec tego rozdziału odczytasz swój pierwszy tag NFC za pomocą urządzenia z systemem Android.

Rozdział 4 to szczegółowy przegląd formatu wymiany danych NFC (NDEF). Dowiesz się, jak jest skonstruowany i zobaczysz go w praktyce, pisząc aplikację, która wykonuje to samo podstawowe zadanie przy użyciu różnych typów rekordów NDEF, aby zobaczyć, jak każdy typ rekordu wpływa na interakcję użytkownika na Androidzie.

Rozdział 5 omawia sposób nasłuchiwania wiadomości NDEF w systemie Android. Dowiesz się, jak filtrować różne typy tagów i wiadomości oraz jak można wykorzystać system Android Tag Dispatch podczas tworzenia aplikacji NFC.

W rozdziale 6 stworzysz pełną aplikację NFC na Androida, która zawiera pełny interfejs użytkownika, odtwarzanie dźwięku i sterowanie oświetleniem podłączonym do sieci, a wszystko to za pośrednictwem tagów NFC. Celem tego rozdziału jest pokazanie, jak zaplanować projekt interakcji i formatowanie danych aplikacji, aby jak najlepiej wykorzystać NFC.

Rozdział 7 wprowadza do gry kolejną platformę: platformę programistyczną mikrokontrolera Arduino. Dowiesz się, jak odczytywać i zapisywać komunikaty NDEF przy użyciu biblioteki Arduino NDEF. Stworzysz także kolejną pełną aplikację przy użyciu Arduino i Node.js.

Rozdział 8 wprowadza do wymiany peer-to-peer przy użyciu NFC w systemie Android. Dowiesz się, w jaki sposób typy rekordów wymienianych przez peer-to-peer wpływają na urządzenie odbierające, a także dowiesz się, w jaki sposób NFC może negocjować przekazywanie większych wymian do innych operatorów, takich jak Bluetooth i WiFi.

Rozdział 9 przedstawia aktualny stan wiedzy na temat rozwoju NFC na wbudowanych platformach Linux na przykładzie Raspberry Pi i BeagleBone. Zrozumiesz, co jest możliwe na wbudowanym Linuksie i zobaczysz kilka przykładowych aplikacji w Node.js. W tym kontekście jest jeszcze wiele miejsca na poprawę użyteczności, więc ostrzegam, że ten rozdział nie jest przeznaczony dla osób nieśmiałych technicznie. Aby jak najlepiej wykorzystać ten rozdział, będziesz potrzebował pewnej znajomości interfejsu wiersza poleceń systemu Linux. Jest to jednak miejsce, w którym kryją się niektóre z najbardziej ekscytujących możliwości wykorzystania NFC, więc warto je znać.

Co będzie potrzebne

Aby wykonać ćwiczenia zawarte w tej książce, będziesz potrzebować sprzętu i oprogramowania. Na szczęście całe oprogramowanie jest darmowe. Najdroższym sprzętem jest urządzenie z Androidem obsługujące NFC. Poniższe sekcje zawierają listę urządzeń, z których będziesz korzystać.

Sprzęt

Aby śledzić książkę w całości, będziesz potrzebować następującego sprzętu:

- Urządzenie z systemem Android i obsługą NFC
- Kilka tagów kompatybilnych z NFC (sprawdź kompatybilność ze swoimi urządzeniami; "Dopasowanie urządzenia do typu tagu" na stronie 19 zawiera tabelę pokazującą, które urządzenia współpracują z poszczególnymi typami tagów)

Do rozdziału 6 potrzebne będą:

- System oświetlenia Philips Hue
- Odbiornik muzyczny Bluetooth (np. Belkin Bluetooth Music Receiver lub HomeSpot Bluetooth Audio Receiver z obsługą NFC)

W przypadku rozdziału 7 będziesz potrzebować:

- Mikrokontroler Arduino Uno, dostępny w wielu sklepach, w tym Arduino, Adafruit, Seeed Studio, RadioShack i innych.
- Tarcza NFC (można użyć tarczy PN532 NFC/RFID Controller Shield dla Arduino firmy Adafruit lub tarczy NFC Shield lub NFC Shield v2.0 firmy Seeed

Studio).

- Jeśli korzystasz z tarczy Adafruit, możesz również zaopatrzyć się w kilka głowic do układania tarcz od Adafruit.

- Elektromagnetyczny zamek do drzwi, 12 V lub mniej. Użyliśmy solenoidu Amico 0837L DC 12V 8W typu open frame do elektrycznego zamka drzwi kupionego na Amazon, ale można również uzyskać solenoidy od innych sprzedawców detalicznych. Adafruit sprzedaje podobny solenoid w stylu zamka, a Seeed Studio sprzedaje kilka modeli, więc jeśli zamawiasz u nich tarczę, możesz również uzyskać od nich solenoid.
- Tranzystor Darlingtona TIP120
- Zasilacz 12 V, 1000 mA, ze złączem 2,1 mm ID, 5,5 mm OD, środkowo-dodatnim do zasilania obwodu elektromagnesu
- Przewody połączeniowe lub przewód solid-core 22AWG
- Dwie diody LED (jedna czerwona, jedna zielona). Są one dostępne w każdym sklepie z elektroniką, ale dla porównania, sprawdź czerwony lub zielony pakiet LED Adafruit.
- Dwa rezystory 220Ω dla diod

LED Do rozdziału 9 potrzebne będą:

- Mikrokontroler BeagleBone Black lub Raspberry Pi z wbudowanym systemem Linux
- Zasilacz o natężeniu 1 A lub większym dla danej płyty
- Bezstykowy czytnik kart inteligentnych USB SLC3711
- Opcjonalne, ale przydatne:
  - Adapter USB WiFi dla twojej płytki (Miniaturowy moduł WiFi (802.11b/g/n) firmy Adafruit działa dobrze).
  - Przedłużacz USB A do A dla adaptera NFC
  - Kabel szeregowy USB do TTL - kabel do debugowania/konsoli

Tabela 1-1 zawiera listę komponentów elektronicznych dla tej książki wraz z numerami części od niektórych dystrybutorów elektroniki, z których regularnie korzystamy, na wypadek gdybyś potrzebował alternatywy dla tych wymienionych wcześniej.

Tabela 1-1. Komponenty elektroniczne użyte w tej książce

Część	MakerShed	Jameco	Digikey	SparkFun	Adafruit	Farnell	Arduino	Seeed
Rezystor 220Ω		690700	220QBK-ND			9337792		
Płytka prototypowa bez lutowania	MKEL3	20723	438-1045-ND	PRT-00137	64	4692810		STR101C2M lub STR102C2M
Czerwony przewód połączeniowy	MKSEEED3	36856	C2117R-100-ND	PRT-08023		1662031		

Czarny przewód połączeniowy	MKSEEED3	36792	C2117B-100-ND	PRT-08022	1662027
Niebieski przewód połączeniowy	MKSEEED3	36767			1662034

	PartMaker	Shed	Jameco	Digikey		SparkFun	Adafruit	Farnell	Arduino Seeed
12V 1000mA DC zasilacz (lub odpowiednik)			170245		TOL-00298	798	636363		
Moduł mikrokontrolera Arduino Uno rev3	MKSP11		2121105	1050-1017-ND	DEV-09950	5 0 1 8 4 8 6 8 7		A000046	ARD132D2P
Zielona dioda LED	MKEE7		333227		COM-09592		1334976		
Czerwona dioda LED	MKEE7		333973		COM-09590		2062463		
Niebieska dioda LED			2006764		COM-00529		1020554		
Żółta dioda LED	MKEE7		34825		COM-09594		1939531		
Znacznik RFID MIFARE					MKPX4SEN-10128				
Osłona NFC	MKAD45					789			SLD01097P
Tranzystor Darlingtona TIP120			10001_ 10001_ 32993_-1	TIP120-ND		976	9294210		
3.3V USB/TTL kabel debugowania szeregowego					DEV-09717	954			
BeagleBone Black	MKCCE3		2176149	BB-BBLK-000-ND		1278	2291620		ARM00100P
Raspberry Pi Model B					MKRPI2DEV-11546	998	43W5302		

Oprogramowanie

Aby śledzić książkę w całości, będziesz potrzebować następującego oprogramowania:

- Zestaw programistyczny Android (patrz "Konfigurowanie środowiska programistycznego" na stronie 24)
- Cordova CLI, zestaw narzędzi dla PhoneGap (patrz "Instalacja Cordova CLI dla PhoneGap" na stronie 28).
- Node.js i menedżer pakietów Node (npm)
- Edytor tekstu (lubimy Sublime Text 2 jako wieloplatformowy edytor GUI, ale można użyć dowolnego, który może wygenerować plik tekstowy).

Do rozdziału 6 potrzebne będą:

- Biblioteka Zepto jQuery, dostępna na stronie

Zepto.js Do rozdziału 7 potrzebne będą:

- Arduino IDE

- Biblioteka Seeed-Studio PN532
- Biblioteka Arduino NDEF, dostępna w repozytorium GitHub Dona Colemana
- Biblioteka Time dla Arduino autorstwa Michaela Margolisa

Nie martw się o konfigurację tego wszystkiego teraz; poinformujemy Cię, kiedy będziesz musiał zainstalować oprogramowanie.

Inne przydatne aplikacje NFC

Poniższe informacje będą przydatne w trakcie całej książki:

- NFC TagInfo firmy NXP umożliwia odczyt dowolnego tagu NFC lub Mifare i sprawdzenie jego rekordu NDEF.
- NFC TagWriter firmy NXP pozwala robić wiele takich samych rzeczy jak TagInfo. Może również zapisywać tagi i niesformatowane tagi, co jest naprawdę przydatne.
- NFC Research Lab's NFC TagInfo pokaże ci wszystkie informacje o danym tagu. Jest bardziej zaawansowany niż TagInfo firmy NXP, ponieważ pozwala również zobaczyć zrzut pamięci z tagu. Jest to nieocenione narzędzie do rozwiązywania problemów z aplikacjami.

Do rozdziału 4 potrzebne będą:

- Wyzwalanie przez TagStand
- NFC Writer, również od TagStand
- TecTiles by Samsung (działa tylko w Stanach Zjednoczonych i Kanadzie)
- App Lancher NFC Tag Writer [sic] by vvakame
- Konto Foursquare, jeśli jeszcze go nie masz, i Foursquare dla Androida

Konwencje stosowane w niniejszej książce

W niniejszej książce zastosowano następujące konwencje typograficzne:

Kursywa
: Wskazuje nowe terminy, adresy URL, adresy e-mail, nazwy plików i rozszerzenia plików.

`Stała szerokość`
: Używany w listach programów, a także w akapitach do odwoływania się do elementów programu, takich jak nazwy zmiennych lub funkcji, bazy danych, typy danych, zmienne środowiskowe, instrukcje i słowa kluczowe.

Ta ikona oznacza wskazówkę, sugestię lub ogólną uwagę.

Ta ikona oznacza ostrzeżenie lub przestrogę.

Korzystanie z przykładów kodu

Materiały uzupełniające (przykłady kodu, ćwiczenia itp.) są dostępne do pobrania pod adresem
https://github.com/tigoe/beginningnfc.

Ta książka ma na celu pomóc w wykonaniu zadania. Ogólnie rzecz biorąc, jeśli przykładowy kod jest oferowany wraz z tą książką, możesz go używać w swoich programach i dokumentacji. Nie musisz kontaktować się z nami w celu uzyskania zgody, chyba że powielasz znaczną część kodu. Na przykład napisanie programu, który wykorzystuje kilka fragmentów kodu z tej książki, nie wymaga zgody. Sprzedaż lub dystrybucja płyt CD-ROM z przykładami z książek O'Reilly wymaga zgody. Udzielenie odpowiedzi na pytanie, powołując się na tę książkę i cytując przykładowy kod, nie wymaga zezwolenia. Włączenie znacznej ilości przykładowego kodu z tej książki do dokumentacji produktu wymaga zezwolenia.

Doceniamy atrybucję, ale jej nie wymagamy. Atrybucja zwykle obejmuje tytuł, autora, wydawcę i numer ISBN. Na przykład: "*Beginning NFC: Near Field Communication with Arduino, Android, and PhoneGap*", autorstwa Toma Igoe, Dona Colemana i Briana Jepsona. Copyright Tom Igoe, Don Coleman, and Brian Jepson 2014 978-1-4493-6307-9."

Jeśli uważasz, że wykorzystanie przykładów kodu wykracza poza dozwolony użytek lub powyższe zezwolenie, skontaktuj się z nami pod adresem *permissions@oreilly.com.*

Safari® Książki online

Safari Books Online to cyfrowa biblioteka na żądanie, która dostarcza treści eksperckie w formie książek i filmów od wiodących światowych autorów w dziedzinie technologii i biznesu.

Specjaliści technologiczni, programiści, projektanci stron internetowych oraz profesjonaliści biznesowi i kreatywni korzystają z Safari Books Online jako głównego

źródła badań, rozwiązywania problemów, nauki i szkoleń certyfikacyjnych.

Safari Books Online oferuje szeroką gamę produktów i programów cenowych dla organizacji, agencji rządowych i osób prywatnych. Subskrybenci mają dostęp do tysięcy książek, filmów szkoleniowych i manuskryptów przed publikacją w jednej w pełni przeszukiwalnej bazie danych od wydawców takich jak O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, Adobe Press, FT Press, Syngress. Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technol- gy i dziesiątki innych. Aby uzyskać więcej informacji na temat Safari Books Online, odwiedź nas online.

Jak się z nami skontaktować

Komentarze i pytania dotyczące tej książki należy kierować do wydawcy:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (w Stanach Zjednoczonych lub
Kanadzie) 707-829-0515 (międzynarodowo
lub lokalnie)
707-829-0104 (faks)

Mamy stronę internetową dla tej książki, na której zamieszczamy erraty, przykłady i wszelkie dodatkowe informacje. Dostęp do tej strony można uzyskać pod adresem http://oreil.ly/beginning-nfc.

Aby skomentować lub zadać pytania techniczne dotyczące tej książki, wyślij wiadomość e-mail na adres *bookques* tions@oreilly.com.

Więcej informacji o naszych książkach, kursach, konferencjach i nowościach można znaleźć na naszej stronie internetowej *http://www.oreilly.com.*

Znajdź nas na Facebooku: *http://facebook.com/oreilly*

Śledź nas na Twitterze: *http://twitter.com/oreillymedia*

Oglądaj nas na YouTube: *http://www.youtube.com/oreillymedia*

Podziękowania

Podczas pisania tej książki otrzymaliśmy hojną pomoc od wielu osób i organizacji. Wtyczka PhoneGap-NFC była pomysłem Kevina Griffina; on i Don napisali jej pierwszą wersję i zaprezentowali ją na PhoneGap Day 2011. Kevin Town- send z Adafruit był nieocenionym źródłem dogłębnej wiedzy na temat oprogramowania i sprzętu NXP. Yihui Xiong z Seeed Studio i autor biblioteki Seeed Arduino NDEF był

kluczowy dla sukcesu rozdziału 7. Philippe Teuwen wprowadził szybkie poprawki do libfreefare, usuwając blokady w rozdziale 9. Strony Dereka Molloya poświęcone BeagleBone

Black były przydatnym źródłem informacji. Zdjęcia w tej książce są lepsze dzięki Jody Culkin i Fritzing.org. Pierwsi czytelnicy: Ben Light, Sae Huh, Gabrielle Levine, Alex Kauffmann, Fil Maj i Dominick Gruntz przekazali cenne uwagi.

Wydanie tej książki było możliwe dzięki cierpliwemu wsparciu naszych pracodawców i kolegów z NYU's Interactive Telecommunications Program (ITP), Chariot Solutions LLC, Maker Media i Arduino. Chcielibyśmy również podziękować za jeszcze bardziej cierpliwe wsparcie naszych rodzin i partnerów.

ROZDZIAŁ 2

NFC i RFID

Identyfikacja radiowa (RFID) staje się obecnie powszechna w codziennym życiu. Od kart płatniczych typu "dotknij i idź" i przepustek tranzytowych, przez urządzenia E-ZPass używane na płatnych drogach, po tagi przyklejane i wszywane do towarów konsumpcyjnych w celu zarządzania zapasami i zapobiegania kradzieży, większość z nas spotyka się z tagami RFID co najmniej kilka razy w tygodniu i nigdy nie myśli o tym, co można zrobić z tą technologią.

W ciągu ostatnich kilku lat pojawił się nowy termin związany z RFID: *komunikacja bliskiego zasięgu* (NFC). Zapytaj przeciętnego technika, co to jest, a prawdopodobnie usłyszysz: "Och, to jak RFID, tylko inaczej". Świetnie, ale czym to się różni? RFID i NFC są często mylone, ale to nie to samo. Chociaż czytniki NFC mogą odczytywać i zapisywać dane na niektórych tagach RFID, NFC ma więcej możliwości niż RFID i umożliwia szerszy zakres zastosowań. Można myśleć o NFC jako o rozszerzeniu RFID, opierając się na kilku z wielu standardów RFID w celu stworzenia szerszej platformy wymiany danych.

Niniejsza książka ma na celu praktyczne zapoznanie użytkownika z technologią NFC i jej możliwościami. Wykonując ćwiczenia zawarte w tych rozdziałach, stworzysz kilka aplikacji NFC dla urządzenia z systemem Android obsługującego NFC oraz dla mikrokontrolera Arduino. Dowiesz się, gdzie RFID i NFC nakładają się na siebie i co można zrobić z NFC.

Co to jest RFID?

Wyobraź sobie, że siedzisz w nocy na werandzie. Włączasz światło na werandzie i możesz zobaczyć swojego sąsiada, gdy przechodzi w pobliżu twojego domu, ponieważ światło odbija się od niego z powrotem do twoich oczu. To właśnie jest pasywne RFID. Sygnał radiowy z pasywnego czytnika RFID dociera do tagu, tag pochłania energię i "odbija" swoją tożsamość.

Teraz wyobraź sobie, że włączasz światło na werandzie, a twój sąsiad w swoim domu

widzi to i włącza światło na werandzie, abyś mógł zobaczyć, jak macha na powitanie ze swojej werandy. To właśnie jest aktywna technologia RFID. Może mieć większy zasięg, ponieważ odbiornik ma własne źródło zasilania,

i dlatego może generować własny sygnał radiowy, zamiast polegać na energii pochłanianej przez nadawcę.

RFID jest jak te dwa ganki. Ty i twój sąsiad znacie swoje twarze, ale tak naprawdę nie dowiadujecie się wiele o sobie nawzajem. Nie wymieniacie żadnych złośliwych wiadomości. RFID nie jest technologią komunikacyjną; jest raczej przeznaczony do identyfikacji. Tagi RFID mogą przechowywać niewielką ilość danych i można je odczytywać i zapisywać za pomocą czytników RFID, ale ilość danych, o których mówimy, jest trywialna, tysiąc bajtów lub mniej.

Co to jest NFC?

Wyobraź sobie teraz, że w pobliżu przechodzi inna sąsiadka, a gdy ją widzisz, zapraszasz ją na werandę na pogawędkę. Przyjmuje zaproszenie i siadacie razem, wymieniacie uprzejmości na temat swojego życia i nawiązujecie bliższą relację. Rozmawiacie ze sobą i słuchacie siebie nawzajem przez kilka minut. To jest właśnie NFC.

Technologia NFC została zaprojektowana w oparciu o RFID, umożliwiając bardziej złożoną wymianę między uczestnikami. Nadal można odczytywać pasywne tagi RFID za pomocą czytnika NFC i można zapisywać dane w ich ograniczonej ilości pamięci. NFC pozwala również na zapisywanie danych do niektórych typów tagów RFID przy użyciu standardowego formatu, niezależnie od typu tagu. Można również komunikować się z innymi urządzeniami NFC w dwukierunkowej wymianie danych. Urządzenia NFC mogą wymieniać się informacjami o swoich możliwościach, wymieniać rekordy i inicjować długoterminową komunikację za pomocą innych środków.

Na przykład, możesz zbliżyć swój telefon z obsługą NFC do zestawu stereo z obsługą NFC, aby oba urządzenia mogły się wzajemnie zidentyfikować, dowiedzieć się, że oba mają łączność Wi-Fi i wymienić dane uwierzytelniające do komunikacji przez Wi-Fi. Następnie telefon zacznie przesyłać strumieniowo dźwięk przez Wi-Fi do zestawu stereo. Dlaczego telefon nie przesyła strumieniowo dźwięku przez połączenie NFC? Z dwóch powodów: po pierwsze, połączenie NFC ma celowo krótki zasięg, zazwyczaj 10 cm lub mniej. Pozwala to na niską moc i uniknięcie zakłóceń z innymi radiami wbudowanymi w urządzenia, które z niego korzystają. Po drugie, jest to stosunkowo niska prędkość w porównaniu do WiFi, Bluetooth i innych protokołów komunikacyjnych. NFC nie jest przeznaczony do zarządzania bardzo szybką komunikacją. Służy do przesyłania krótkich wiadomości, wymiany danych uwierzytelniających i inicjowania relacji. Wróćmy na chwilę na ganek. NFC to zmiana, którą musisz wprowadzić, aby rozpocząć rozmowę. Jeśli chcesz porozmawiać dłużej, zapraszasz sąsiada do środka na herbatę. To jest WiFi, Bluetooth i inne rozszerzone protokoły komunikacyjne.

Ekscytujące w NFC jest to, że pozwala na zaawansowane wprowadzanie i krótkie instrukcje bez kłopotów z wymianą haseł, parowaniem i wszystkimi innymi bardziej

skomplikowanymi krokami, które towarzyszą tym innym protokołom. Oznacza to, że gdy ty i twój przyjaciel chcecie wymienić informacje adresowe z twojego telefonu na jego, to

można po prostu zetknąć ze sobą telefony. Gdy chcesz zapłacić za pomocą Portfela Google, możesz po prostu dotknąć, tak jak w przypadku karty kredytowej z obsługą RFID.

Gdy korzystasz z NFC, Twoje urządzenie nie daje drugiemu urządzeniu, z którym rozmawia, dostępu do całej swojej pamięci - daje mu tylko podstawy potrzebne do wymiany. Ty kontrolujesz, co może wysyłać, a czego nie i do kogo.

Jak działa RFID

Wymiana RFID obejmuje dwóch uczestników: *cel* i *inicjatora*. Inicjator, czytnik tagów lub urządzenie czytająco-zapisujące, rozpoczyna wymianę, generując pole radiowe i oczekując odpowiedzi od dowolnego celu w polu. Obiekt docelowy, tag, odpowiada, gdy odbierze transmisję od inicjatora. W odpowiedzi otrzymuje *unikalny numer identyfikacyjny* (UID). RFID ma dwa *tryby komunikacji*: aktywny i pasywny. *Pasywne* zmiany *RFID* obejmują czytnik/zapis i tag, który nie ma źródła zasilania na pokładzie. Znaczniki pobierają energię z samego pola radiowego. Zazwyczaj jest to bardzo niewielka ilość, wystarczająca do wysłania sygnału z powrotem do czytnika. *Aktywna* wymiana *RFID* obejmuje cel, który jest niezależnie zasilanym urządzeniem. Ponieważ cel jest zasilany, jego odpowiedź do czytnika może przebyć znacznie większą odległość. E-ZPass i inne systemy identyfikacji ruchu wykorzystują aktywną technologię RFID.

Tagi RFID mają niewielką ilość pamięci na pokładzie, zwykle mniej niż 1 kilobajt. Urządzenie inicjujące może odczytywać te dane, a jeśli jest to urządzenie czytające/zapisujące, może również zapisywać na tagu. Pozwala to na przechowywanie niewielkich ilości informacji powiązanych z kartą. Na przykład, jest to czasami używane w systemach tranzytowych, które wykorzystują RFID, aby śledzić, ile wartości pozostało na karcie. Ponieważ jednak systemy RFID są zazwyczaj połączone z bazą danych, bardziej powszechne jest przechowywanie rekordu danych indeksowanego przez identyfikator UID tagu w zdalnej bazie danych i przechowywanie wszystkich informacji o tagu w tej zdalnej bazie danych.

Standardy RFID

Wbrew powszechnemu przekonaniu, nie istnieje jeden powszechnie interoperacyjny protokół lub technologia RFID. Istnieją ich dziesiątki. Standardy RFID są opracowywane przez Międzynarodową Organizację Normalizacyjną (ISO) we współpracy z głównymi uczestnikami rynku RFID. ISO działa jako organ pośredniczący, pomagając konkurentom z wielu różnych branż w opracowywaniu interoperacyjnych standardów, dzięki czemu nawet gdy konkurują, ich technologie mogą czasami współpracować. Różne standardy RFID definiują wykorzystywane częstotliwości radiowe, szybkości transferu danych, formaty danych i inne. Niektóre z tych standardów definiują warstwy jednego interoperacyjnego stosu, jak w przypadku NFC. Inne standardy definiują zupełnie inną klasę aplikacji. Na przykład standard

ISO-11784 został pierwotnie opracowany do śledzenia zwierząt. Działa na częstotliwościach od 129 do 139,4 kHz, a jego format danych zawiera pola odpowiednie do opisywania śledzonych zwierząt. Można również znaleźć czytniki i tagi procotol EM4100, które działają w zakresie 125 kHz. Są one często używane jako karty zbliżeniowe i zawierają bardzo ograniczone informacje w formacie danych.

ich protokół danych, zwykle tylko UID. Standardy ISO-14443 zostały opracowane do użytku z systemami płatności i kartami inteligentnymi. Działają one na częstotliwości 13,56 MHz. Obejmują one funkcje w swoim formacie danych do inkrementacji i dekrementacji wartości oraz do szyfrowania danych, na przykład. W ramach rodziny 14443 istnieje kilka różnych formatów, w tym tagi Philips i NXP Mifare, tagi Sony FeliCa i NXP DESFire. Tagi ISO-14443A są kompatybilne z NFC, więc zobaczysz ich wiele na kolejnych stronach.

Jak działa NFC

NFC można traktować jako rozszerzenie RFID. Wymiana NFC również obejmuje inicjator i cel, podobnie jak RFID. Może jednak robić więcej niż tylko wymieniać identyfikatory UID i odczytywać lub zapisywać dane do celu. Najbardziej interesującą różnicą między RFID i NFC jest to, że cele NFC są często programowalnymi urządzeniami, takimi jak telefony komórkowe. Oznacza to, że zamiast po prostu dostarczać statyczne dane z pamięci, cel NFC może faktycznie generować unikalną zawartość dla każdej wymiany i dostarczać ją z powrotem do inicjatora. Na przykład, jeśli używasz NFC do wymiany danych adresowych między dwoma telefonami, docelowe urządzenie NFC można zaprogramować tak, aby dostarczało tylko ograniczone informacje, jeśli nigdy wcześniej nie widziało tego konkretnego inicjatora.

Urządzenia NFC mają dwa *tryby komunikacji*. Jeśli inicjator zawsze dostarcza energię RF, a cel jest zasilany przez pole inicjatora, mówi się, że angażują się w *pasywny* tryb komunikacji. Jeśli zarówno cel, jak i inicjator mają własne źródła energii, są w trybie *aktywnej komunikacji.* Tryby te są takie same jak zwykłe tryby komunikacji RFID.

Urządzenia NFC mają trzy *tryby pracy*. Mogą być *czytnikami/zapisywarkami,* które odczytują dane z celu i zapisują je na nim. Mogą być *emulatorami kart*, działającymi jak tagi RFID, gdy znajdują się w polu innego urządzenia NFC lub RFID. Mogą też działać w *trybie peer-to-peer,* w którym wymieniają dane w obu kierunkach.

Format wymiany danych NFC (NDEF)

Dane wymieniane między urządzeniami NFC i tagami są formatowane przy użyciu formatu *NFC Data Ex- change Format* (NDEF). Jest to termin, który będzie często słyszany; NDEF jest jednym z kluczowych udogodnień, które NFC dodaje do RFID. Jest to wspólny format danych, który działa na wszystkich urządzeniach NFC, niezależnie od bazowego tagu lub technologii urządzenia. Każda wiadomość NDEF zawiera jeden lub więcej rekordów NDEF. Każdy rekord ma określony typ rekordu, unikalny identyfikator, długość i ładunek danych. Istnieje kilka dobrze znanych typów rekordów NDEF. Oczekuje się, że urządzenia obsługujące NFC będą wiedziały, co zrobić z każdym z tych typów:

Proste rekordy tekstowe
: Zawierają one dowolny ciąg tekstowy, który chcesz wysłać. Wiadomości tekstowe zazwyczaj nie zawierają instrukcji dla urządzenia docelowego. Zawierają również metadane wskazujące język i schemat kodowania (np. UTF-8).

URI
: Zawierają one adresy sieciowe. Oczekuje się, że urządzenie docelowe NDEF, które otrzyma rekord URI, przekaże ten rekord do aplikacji, która może go wyświetlić, takiej jak przeglądarka internetowa.

Inteligentne plakaty
: Zawierają one dane, które można dołączyć do plakatu, aby nadać mu więcej informacji. Może to obejmować identyfikatory URI, ale może również zawierać inne dane, takie jak wiadomość tekstowa do wysłania na temat plakatu, informująca o nim znajomych. Urządzenie docelowe, które otrzyma rekord Smart Poster, może otworzyć przeglądarkę, SMS lub aplikację e-mail, w zależności od treści wiadomości.

Podpisy
: Zapewniają one sposób na podanie wiarygodnych informacji o pochodzeniu danych zawartych w rekordzie NDEF.

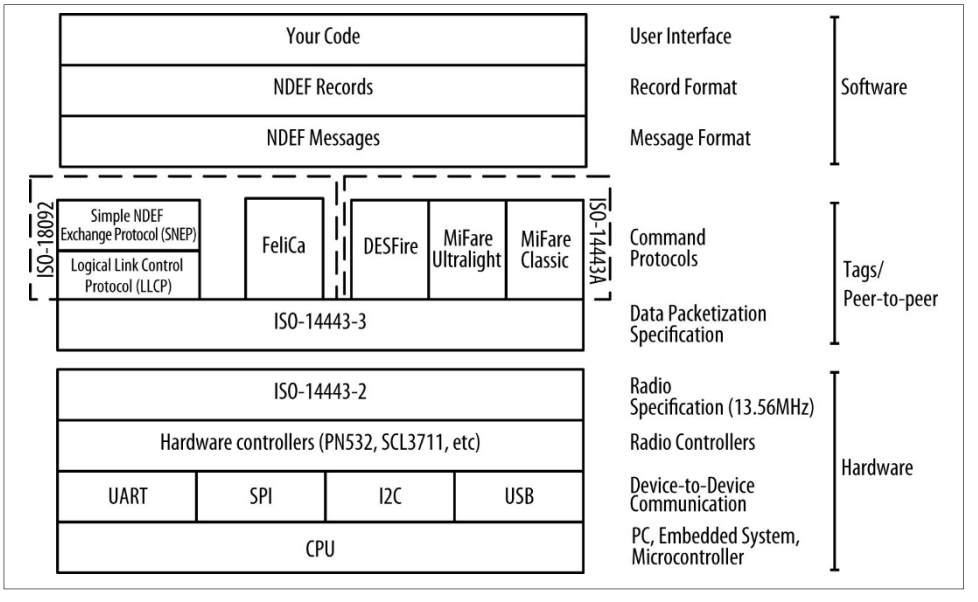
W wiadomości NDEF można łączyć i dopasowywać rekordy lub wysyłać tylko jeden rekord na wiadomość.

NDEF jest jedną z ważnych różnic technicznych między RFID i NFC. Zarówno NFC, jak i wiele protokołów RFID działa na częstotliwości 13,56 MHz, ale tagi RFID nie muszą formatować swoich danych w formacie NDEF. Różne protokoły RFID nie mają wspólnego formatu danych.

Pomyśl o wiadomościach NDEF jak o paragrafach, a o rekordach jak o zdaniach: paragraf to dyskretny fragment informacji, który zawiera jedno lub więcej zdań. Zdanie to mniejszy fragment informacji, który zawiera tylko jedną myśl. Na przykład, możesz napisać akapit, który wskazuje, że organizujesz przyjęcie urodzinowe i podaje adres. Odpowiednik wiadomości NDEF może zawierać prosty rekord tekstowy opisujący wydarzenie, rekord Smart Poster zawierający adres fizyczny oraz URI, aby uzyskać więcej informacji w Internecie.

Architektura NFC

Aby dogłębnie zrozumieć NFC, warto mieć mentalny model architektury. Należy wziąć pod uwagę kilka warstw. Najniższą warstwą jest warstwa fizyczna, a mianowicie procesor i radia, które wykonują komunikację. Pośrodku znajdują się warstwy pakowania i transportu danych, następnie warstwy formatu danych, a na końcu kod aplikacji. Rysunek 2-1 przedstawia różne warstwy stosu NFC.



Rysunek 2-1. Stos protokołów NFC

W warstwie fizycznej NFC działa w oparciu o specyfikację radiową RFID, ISO-14443-2, która opisuje radia o niskiej mocy działające na częstotliwości 13,56 MHz. Następnie pojawia się warstwa opisująca ramkowanie bajtów danych wysyłanych przez radio, ISO-14443-3. Wszystkie używane radia są oddzielnymi komponentami sprzętowymi, znajdującymi się w telefonie lub tablecie, lub dołączonymi do mikrokontrolera lub komputera osobistego. Komunikują się one z głównym procesorem urządzenia za pomocą jednego lub kilku standardowych protokołów szeregowych między urządzeniami: *uniwersalnego asynchronicznego odbioru i nadawania* (UART), *szeregowego interfejsu peryferyjnego* (SPI), *komunikacji między układami scalonymi* (I2C) lub *uniwersalnej magistrali szeregowej* (USB). Jeśli jesteś entuzjastą sprzętu komputerowego, prawdopodobnie znasz trzy pierwsze, ale jeśli koncentrujesz się na oprogramowaniu, możesz znać tylko USB.

Powyżej znajduje się kilka protokołów poleceń RFID, opartych na dwóch specyfikacjach. Oryginalna specyfikacja sterowania RFID, na której opiera się odczyt i zapis tagów NFC, to ISO-14443A. Protokoły Philips/NXP Semiconductors Mifare Classic i Mifare Ultralight oraz NXP DESFire są zgodne z ISO-14443A. Wymiana peer-to-peer NFC jest oparta na protokole sterowania ISO-18092. Karty i tagi Sony FeliCa RFID, dostępne głównie w Japonii, są również oparte na tym standardzie. Można jednak odczytywać i zapisywać na tagach przy użyciu tych standardów i nadal nie korzystać z NFC.

Pierwszą główną różnicą między NFC a RFID jest tryb komunikacji peer-to-peer, który jest realizowany przy użyciu standardu ISO-18092. Istnieją dwa protokoły, protokół *kontroli łącza logicznego (*LLCP) i *prosty protokół wymiany NDEF* (SNEP),

które

zarządzają wymianą peer-to-peer. Są one pokazane na stosie obok protokołów kontrolnych, ponieważ działają w tym samym standardzie, co jeden z nich.

Druga główna różnica między RFID i NFC, format wymiany danych NFC (NDEF), znajduje się na szczycie tych protokołów kontrolnych. NDEF definiuje wymianę danych w komunikatach, które składają się z rekordów NDEF. Istnieje kilka różnych typów rekordów, o których dowiesz się więcej w rozdziale 4. NDEF umożliwia kodowi aplikacji obsługę danych z odczytu i zapisu tagów NFC, wymiany peer-to-peer i emulacji karty w ten sam sposób.

Większość tej książki koncentruje się na wykorzystaniu komunikatów NDEF. Chociaż omówimy niższe warstwy, abyś je zrozumiał, uważamy, że zaletą NDEF jest to, że uwalnia cię od myślenia o nich.

Typy tagów NFC

Istnieją cztery typy tagów zdefiniowane przez forum NFC, wszystkie oparte na protokołach kontroli RFID opisanych wcześniej. Jest jeszcze piąty, który jest kompatybilny, ale nie jest ściśle częścią specyfikacji NFC. Typy 1, 2 i 4 są oparte na ISO-14443A, a typ 3 jest oparty na ISO-18092. Tagi można nabyć od wielu dostawców i prawdopodobnie można się z nimi zetknąć w życiu codziennym, nie wiedząc o tym. Ich szczegóły są następujące:

Typ 1

- Na podstawie specyfikacji ISO-14443A.
- Może być tylko do odczytu lub do odczytu/zapisu.
- 96 bajtów do 2 kilobajtów pamięci.
- Prędkość komunikacji 106Kb.
- Brak ochrony przed kolizją danych.
- Przykłady: Innovision Topaz, Broadcom BCM20203.

Typ 2

- Podobnie jak znaczniki typu 1, znaczniki typu 2 są oparte na specyfikacji NXP/Philips Mifare Ultralight (ISO-14443A).
- Może być tylko do odczytu lub do odczytu/zapisu.
- 96 bajtów do 2 kilobajtów pamięci.
- Prędkość komunikacji 106Kb.
- Wsparcie antykolizyjne.
- Przykład: NXP Mifare Ultralight.

Typ 3

- Są one oparte na znacznikach Sony FeliCa (ISO-18092 i JIS-X-6319-4), bez obsługi szyfrowania i uwierzytelniania, które zapewnia FeliCa.

- Fabrycznie skonfigurowany jako tylko do odczytu lub do odczytu/zapisu.
- Zmienna pamięć, do 1 MB na wymianę.
- Dwie prędkości komunikacji, 212 lub 424 kb/s.
- Wsparcie antykolizyjne.
- Przykład: Sony FeliCa.

Typ 4

- Podobnie jak znaczniki typu 1, znaczniki typu 4 są oparte na specyfikacji NXP DESFire (ISO-14443A).
- Fabrycznie skonfigurowany jako tylko do odczytu lub do odczytu/zapisu.
- 2, 4 lub 8 KB pamięci.
- Zmienna pamięć, do 32 KB na wymianę.
- Trzy prędkości komunikacji: 106, 212 lub 424Kbps.
- Wsparcie antykolizyjne.
- Przykład: NXP DESFire, SmartMX-JCOP.

Piąty typ, zastrzeżony przez NXP Semiconductors, jest prawdopodobnie najbardziej powszechny w dzisiejszych zastosowaniach NFC:

Znacznik Mifare Classic (ISO-14443A)

- Opcje pamięci: 192, 768 lub 3584 bajty.
- Prędkość komunikacji 106Kbps
- Wsparcie antykolizyjne.
- Przykłady: NXP Mifare Classic 1K, Mifare Classic 4K, Mifare Classic Mini.

Gdzie zdobyć tagi

Znaczniki Mifare i NFC są coraz łatwiejsze do zdobycia. Aby w pełni wykorzystać tę książkę, powinieneś mieć kilka tagów NFC Forum i kilka tagów Mifare Classic.

Wiele sklepów z telefonami komórkowymi oferuje obecnie kafelki TecTiles firmy Samsung, które dobrze sprawdzają się w wielu aplikacjach opisanych w tej książce. Są drogie, ale wygodne. Oryginalne tagi TecTile to Mifare Classic, które działają dobrze na Arduino w rozdziale 7 i czytniku USB NFC na BeagleBone Black i Raspberry Pi w rozdziale 9. Nie jest to jasne, ale wydaje się, że Samsung porzuca TecTile na rzecz TecTile 2. Producenci sprzętu odchodzą od Mifare Classic na rzecz oficjalnych typów tagów NFC Forum.

Jeśli potrzebujesz tagów w różnych formatach, Adafruit ma szeroki wybór w swojej kategorii NFC. SparkFun i Seeed Studio mają również wiele tagów Mifare; wyszukaj "NFC" lub "Mifare", a znajdziesz wiele.

Tagi można również uzyskać od TagAge, ale wysyłka z Finlandii może być powolna, w zależności od lokalizacji. Obsługa klienta była jednak dla nas doskonała.

Identive NFC może dostarczyć tagi z dnia na dzień. Są nieco droższe, ale docierają szybko i mają fajnego androida.

Dopasowanie urządzenia do typu znacznika

Nie wszystkie tagi kompatybilne z NFC działają ze wszystkimi urządzeniami NFC. W trakcie pracy nad tą książką przetestowaliśmy wiele różnych urządzeń i tagów i odkryliśmy, że niektóre kombinacje działają, a niektóre nie. Chociaż tabela 2-1 nie jest wyczerpująca, ma na celu dostarczenie wystarczającej ilości informacji, aby odpowiednio wybrać urządzenia i tagi.

Najczęstszym problemem jest to, że niektóre urządzenia odczytują tagi Mifare Classic, podczas gdy inne nie. Mifare Classic nie jest w rzeczywistości standardowym typem tagu NFC Forum, a ostatnio główni producenci sprzętu, tacy jak Broadcom i Samsung, zaczęli rezygnować z jego obsługi w swoich urządzeniach. Prawdopodobnie uzyskasz nieprzewidywalne wyniki, gdy spróbujesz odczytać tagi Mifare Classic za pomocą niektórych z tych nowszych tagów. Na przykład Nexus 4 odczyta Mifare Classic jako tag ogólny, ale Samsung S4 (Google Edition) wyświetli błąd informujący, że nie może obsłużyć tagu. Jeśli to możliwe, zalecamy korzystanie z jednego z oficjalnych typów tagów NFC Forum. W kolejnych rozdziałach przedstawimy bardziej szczegółowe zalecenia w stosownych przypadkach.

Jeśli przechodzisz do NFC, ponieważ jesteś zainteresowany wykorzystaniem go w projektach sprzętu wbudowanego na Arduino, Raspberry Pi lub BeagleBone Black (o których dowiesz się więcej w rozdziałach 7 i 9), powinieneś upewnić się, że masz zarówno tagi NFC, jak i tagi Mifare Classic. Biblioteki dla tych platform zostały opracowane począwszy od obsługi Mi- fare Classic i w chwili pisania tego tekstu nie obsługują wszystkich typów NFC Forum. Dodaliśmy obsługę odczytu Mifare Ultralight (typ NFC 2) do biblioteki Arduino NDEF (którą napisał Don), a częściowa obsługa Ultralight i DESFire (typy NFC 2 i 4) znajduje się w libfreefare, jednej z bibliotek, które pokażemy w rozdziale 9.

Chociaż mamy nadzieję, że zobaczymy dodatki do tych bibliotek, aby obsługiwać wszystkie typy tagów NFC Forum, to jeszcze nie teraz. Bóle wzrostowe, takie jak te, są powszechne w przypadku nowych technologii, ale jak dotąd znaki wskazują, że branża zmierza w kierunku znormalizowanego zestawu typów tagów.

Tabela 2-1. Kompatybilność urządzeń z tagami

Urządzenie	Typ 1	Typ 2 (Mifare Ultralight)	Typ 3 (Sony FeliCa)	Typ 4 (Mifare DESFire)	Mifare Classic
Samsung Galaxy S	Tak	Tak	Tak	Tak	Tak
Google Nexus S	Tak	Tak	Tak	Tak	Tak
Google Galaxy Nexus	Tak	Tak	Tak	Tak	Tak
Google Nexus 7 wersja 1	Tak	Tak	Tak	Tak	Tak
Google Nexus 7 wersja 2	Tak	Tak	Tak	Tak	Nie
Google Nexus 4 (telefon)	Tak	Tak	Tak	Tak	Nie
Google Nexus 10 (tablet)	Tak	Tak	Tak	Tak	Nie
Samsung Galaxy S4 (tablet)	Tak	Tak	Tak	Tak	Nie
Samsung Galaxy SIII	Tak	Tak	Tak	Tak	Tak
Tarcza NFC Adafruit	Nie	Częściowy	Nie	Nie	Tak
Seeed Studio NFC Shield dla Arduino	Nie	Częściowy	Nie	Nie	Tak
Klucz sprzętowy USB NFC (libnfc)	Nie	Częściowy	Nie	Częściowy	Tak

Co można zrobić z NFC

Istnieje wiele możliwych zastosowań NFC. Jego funkcja emulacji tagów pojawia się w transakcjach pieniężnych, takich jak Google Wallet, oraz w systemach płatności i sprzedaży biletów w transporcie publicznym. Istnieje kilka aplikacji na telefony komórkowe, które pozwalają na zapisanie danych konfiguracyjnych telefonu w tagu i użycie tagu do automatycznej zmiany kontekstu telefonu (np. ustawienie ciszy podczas spotkań, włączenie WiFi i połączenie z daną siecią poprzez dotknięcie tagu). Na rynku pojawia się domowy sprzęt audio, który umożliwia automatyczne parowanie telefonu lub tabletu z odtwarzaczem audio lub telewizorem, aby używać tego pierwszego jako pilota do tego drugiego. Systemy opieki zdrowotnej wdrożyły identyfikację pacjentów i łączenie rekordów za pomocą NFC, a NFC Forum niedawno wydało specyfikację techniczną Personal Health Device Communication (PHDC), aby pomóc w tym rozwoju. Zarządzanie zapasami może zostać usprawnione poprzez wykorzystanie NDEF do przesyłania informacji o wysyłanych towarach, ich podróży, czasie wysyłki itp. Śledzenie floty można również usprawnić za pomocą tagów i czytników NFC, przechowując informacje o trasie, czasie podróży, przebiegu, konserwacji i paliwie na tagach lub przenosząc je z pojazdu na urządzenie mobilne za pośrednictwem peer-to-peer.

W trakcie pracy nad tą książką widzieliśmy tagi kompatybilne z NFC w wielu miejscach, od breloczków Citi Bike w Nowym Jorku, przez bilety metra w Chinach i Norwegii, po klucze do pokoi hotelowych i nie tylko. Chociaż nie wszystkie z tych aplikacji wykorzystują pełny potencjał NFC, wszystkie mogą zostać ulepszone lub uproszczone dzięki niektórym z jego funkcji.

Połączenie przesyłania wiadomości peer-to-peer między urządzeniami przy minimalnych negocjacjach oraz wspólnego formatu danych między wiadomościami urządzenia i tagu sprawia, że NFC jest potencjalnie potężnym narzędziem do zastosowań wymagających dużej ilości danych w świecie fizycznym.

Wnioski

Komunikacja bliskiego zasięgu dodaje kilka obiecujących nowych funkcji do technologii RFID. Najważniejszym z nich jest NFC Data Exchange Format, NDEF, który zapewnia wspólny format danych dla czterech różnych technologii tagów NFC. NDEF może być używany zarówno do wymiany danych między tagami, jak i między urządzeniami. Wyróżnia to NFC jako coś więcej niż tylko technologię identyfikacji; jest to również przydatna technologia do krótkiej wymiany danych. Metoda, za pomocą której urządzenia i tagi NFC wchodzą w interakcję - za pomocą pojedynczego dotknięcia - zapewnia spontaniczność interakcji między użytkownikami.

Istnieją cztery fizyczne typy tagów NFC. Są one oparte na istniejących standardach RFID; trzy typy tagów są oparte na standardzie ISO-14443A, a czwarty jest oparty na standardzie ISO-18092. Dzięki temu tagi NFC są przynajmniej częściowo kompatybilne z wieloma istniejącymi systemami RFID Mifare i FeliCa. Chociaż te starsze systemy nie obsługują NDEF, nadal mogą identyfikować te tagi NFC, które są z nimi kompatybilne. Na przykład czytnik RFID, który może odczytywać tagi Mifare Ultralight, nadal będzie w stanie odczytać identyfikator UID tagu NFC typu 2, nawet jeśli nie będzie w stanie odczytać żadnych danych NDEF zakodowanych na tagu.

W obecnym stanie istnieją pewne niezgodności między niektórymi urządzeniami NFC a starszymi tagami Mifare Classic. Niektóre radia NFC obsługują Mifare Classic jako nieoficjalny piąty typ tagu, podczas gdy inne nie. Miejmy nadzieję, że w miarę dojrzewania NFC i pojawiania się na rynku większej liczby urządzeń, te niezgodności będą miały mniejszy wpływ na użytkowników. Na razie jednak najlepiej jest sprawdzić kompatybilność między czytnikami i tagami, których planujesz używać w danej aplikacji.

ROZDZIAŁ 3

Pierwsze kroki z PhoneGap i Biblioteka PhoneGap-NFC

PhoneGap to framework programistyczny, który umożliwia tworzenie aplikacji dla systemów iOS, Android, BlackBerry, Windows Phone 7 i 8, Symbian i Bada przy użyciu (głównie) HTML5 i JavaScript. Ludzie z PhoneGap stworzyli coś, co jest zasadniczo podstawową aplikacją przeglądarkową, ale bez interfejsu. Implementujesz interfejs w HTML5 i Java- Script, a następnie kompilujesz aplikację dla danej platformy. Opracowali również system wtyczek, dzięki czemu programiści mogą rozszerzyć podstawową aplikację przeglądarkową przy użyciu natywnych funkcji różnych platform. Jest to przydatne, ponieważ oznacza to, że nie musisz znać natywnych frameworków aplikacji dla różnych platform mobilnych, aby pisać aplikacje, które mogą działać na nich wszystkich.

W tej książce będziesz używać PhoneGap do pisania aplikacji na Androida i wtyczki, która umożliwia dostęp do sprzętu NFC wbudowanego w wiele telefonów z Androidem. Kod służący do nasłuchiwania tagów i interakcji z użytkownikiem zostanie napisany w JavaScript, a interfejs zostanie ułożony w HTML5. Powłoka aplikacji jest napisana w Javie, podobnie jak wtyczka NFC, ale nie będziesz musiał zmieniać żadnego kodu powłoki w projektach opisanych w tej książce.

Dlaczego Android?

PhoneGap-NFC obsługuje również Windows Phone 8, BlackBerry 7 i BlackBerry 10. Wybraliśmy Androida do tej książki, ponieważ ma on największy udział w rynku i najwięcej telefonów NFC, a większość użytkowników wtyczki PhoneGap-NFC to użytkownicy Androida. Jeśli masz telefon z systemem Windows Phone lub BlackBerry, powinieneś być w stanie zmodyfikować niektóre przykłady, aby podążać za nimi. Istnieją jednak pewne ograniczenia. Windows Phone 8 może odczytywać,

zapisywać i udostępniać tagi NFC, ale ma dostęp tylko do danych NDEF. Nie widzi typu tagu, identyfikatorów UID tagu ani żadnych innych metadanych tagu. BlackBerry 10 ma podobne ograniczenia: może odczytywać i udostępniać tagi NFC, ale obecnie nie może ich zapisywać.

PhoneGap-NFC dla BlackBerry 7 obsługuje więcej funkcji NFC niż BlackBerry 10, ale będziesz musiał użyć starszych wersji, aby to działało, ponieważ PhoneGap 3.0 porzucił wsparcie dla BlackBerry 7.

Do napisania kodu można użyć dowolnego edytora tekstu, ale do kompilacji kodu i wdrożenia go na urządzeniu wykorzystany zostanie zestaw programistyczny Android (SDK) i powiązane z nim narzędzia. Poznasz również interfejs wiersza poleceń Cordova (CLI), który jest zestawem narzędzi do kompilowania aplikacji PhoneGap przy użyciu Android SDK. Przyda się także odrobina znajomości interfejsu wiersza poleceń komputera.

Instrukcje zakładają, że pracujesz w interfejsie wiersza poleceń, takim jak Ter- minal na dowolnym komputerze z systemem Linux lub OS X. W przypadku użytkowników systemu Windows korzystających z wiersza poleceń istnieją pewne niewielkie różnice, które omówimy.

Hello, World! Twoja pierwsza aplikacja PhoneGap

Do realizacji projektów opisanych w tym rozdziale potrzebne są:

- Edytor tekstu (jeśli nie masz ulubionego edytora tekstu, zalecamy Sublime Text 2, ponieważ jest to dobry wieloplatformowy edytor kodu).
- Telefon z systemem Android obsługujący technologię NFC (lista dostępnych telefonów znajduje się w tabeli 2-1)
- Kilka tagów NFC (aby uzyskać najlepsze wyniki na wielu urządzeniach, trzymaj się typów NFC Forum; zobacz "Dopasowanie typu urządzenia do tagu" na stronie 19, aby dowiedzieć się więcej o tym, które tagi współpracują z określonymi urządzeniami).
- Zestaw do tworzenia oprogramowania Android
- Cordova CLI, zestaw narzędzi dla PhoneGap ("Instalacja Cordova CLI dla PhoneGap" na stronie 28)
- Node.js i menedżer pakietów Node (npm); użyjesz go do pobrania interfejsu Cordova CLI (patrz "Instalowanie Node.js i npm" na stronie 27).

Konfiguracja środowiska programistycznego

Najpierw należy pobrać Android SDK ze strony Android Developer. Android zaleca pobranie pakietu Android Developers Toolkit (ADT), który zawiera dołączoną wersję środowiska programistycznego Eclipse. W tej książce nie korzystamy jednak z Eclipse, więc możesz po prostu pobrać narzędzia SDK dla swojej platformy, co powinno zająć mniej czasu. Aby to zrobić, kliknij "Użyj istniejącego IDE" na dole strony "Pobierz", a otrzymasz link do pobrania samych narzędzi SDK bez Eclipse.

W systemie OS X lub Linux

Wyodrębnij pakiet Android SDK i umieść go w miejscu na komputerze, w którym będzie można go ponownie znaleźć. Na przykład, jeśli umieścisz go w folderze Aplikacje na OS X, ścieżka do niego będzie */Applications/android-sdk-macosx*.

W systemie Windows

Najpierw należy zainstalować pakiet Java Development Kit (JDK), pobierając go z łącza "Java SE" na stronie internetowej Oracle. Po zainstalowaniu Javy można uruchomić instalator Android SDK.

W systemie Windows potrzebna będzie jeszcze jedna rzecz: Apache Ant, który zarządza procesem kompilacji programów na Androida. Pobierz plik ZIP Ant ze strony Apache Ant, rozpakuj plik na dysk *C:* i zmień nazwę folderu najwyższego poziomu (np. *apache-ant-1.9.1-bin*) na *Ant*. Jeśli zrobiłeś to poprawnie, powinieneś znaleźć podkatalogi, takie jak *C:\Ant\bin* i *C:\Ant\lib*.

W systemie Windows instalator Android SDK zostanie domyślnie umieszczony w ukrytym katalogu w katalogu domowym: *C:\Users\Username\AppData\Local\Android\android-sdk*, co utrudni jego późniejsze odnalezienie. Sugerujemy zmianę na *C:\Users\Username\Android\android-sdk*.

Zainstaluj narzędzia platformy Android

W systemie OS X lub Linux

Otwórz okno Terminala i zmień katalog na katalog SDK (tj. użyj `cd /Applications/android-sdk-macosx`). SDK nie zawiera wersji narzędzi platformy Android, więc należy je zainstalować, wpisując następujące polecenie:

```
$ tools/android update sdk --no-ui
```

W systemie Windows

Jeśli instalator Android SDK nie może znaleźć instalacji Java (częsty problem w 64-bitowym systemie Windows), zamknij instalator i ustaw zmienną środowiskową `JAVA_HOME` zgodnie z instrukcjami w dalszej części tej sekcji. Spróbuj ponownie, a instalator powinien znaleźć instalację Java.

Po zakończeniu instalacji pojawi się możliwość zainstalowania narzędzi. Pozostaw zaznaczone pole wyboru "Start SDK Manager" i kliknij "Finish". Gdy pojawi się menedżer SDK, kliknij "Zainstaluj *n* pakietów" (gdzie *n* to dowolna liczba pakietów oferowanych domyślnie). Jeśli przypadkowo odznaczyłeś pole wyboru przed kliknięciem "Zakończ", otwórz wiersz polecenia, zmień katalog na lokalizację SDK (tj. `cd Android\android-sdk`) i uruchom następujące polecenie:

```
C:\Users\Username\Android\android-sdk>tools\android update sdk --no-ui
```

Spowoduje to aktualizację Android SDK i zainstalowanie najnowszych wersji narzędzi platformy. Możesz zostać poproszony o zaakceptowanie niektórych licencji oprogramowania przed kontynuowaniem. Zajmie to trochę czasu, więc warto poświęcić chwilę na rozciągnięcie się i odświeżenie podczas instalacji.

Następnie należy zmienić zmienną `PATH` dla środowiska, aby zawierała narzędzia Android SDK. Katalog narzędzi obejmuje większość narzędzi do kompilowania, przesyłania i uruchamiania aplikacji na urządzeniu, a w narzędziach platformy znajdziesz ważne narzędzie o nazwie `adb`, którego będziesz używać do celów debugowania później.

W poniższych przykładach, jeśli umieścisz narzędzia w lokalizacji innej niż pokazana, zmień ścieżki w razie potrzeby. Należy również pamiętać, że tworzone są dwie ścieżki: jedna dla podkatalogu *tools*, a druga dla *narzędzi platformy*.

W systemie OS X
lub Linux

Zmień katalog na swój katalog domowy i edytuj *.bash_profile* lub *.profile*. Jeśli plik nie istnieje, możesz go po prostu utworzyć. Dodaj następujące linie do tego pliku:

```
export ANDROID_HOME=/Applications/android-sdk-macosx
export PATH=$PATH:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools
```

Oczywiście należy zmienić ścieżkę *ANDROID_HOME*, jeśli SDK zostało zapisane w innym miejscu niż katalog Aplikacje. Następnie zapisz plik profilu, wyloguj się z Terminala i zaloguj ponownie, aby zresetować zmienną `PATH`.

W systemie Windows

Musisz znaleźć drogę do okna dialogowego "Zaawansowane właściwości systemu":

Windows XP

Otwórz menu Start, kliknij prawym przyciskiem myszy "Mój komputer" i wybierz "Właściwości", a następnie przejdź do zakładki "Zaawansowane".

Windows 7 lub Vista

Otwórz menu Start i kliknij prawym przyciskiem myszy "Komputer". Wybierz "Właściwości", a następnie wybierz "Zaawansowane ustawienia systemu" z listy po lewej stronie.

Windows 8.1

Na pulpicie kliknij prawym przyciskiem myszy lub dotknij i przytrzymaj menu Start, a następnie wybierz "System". Wybierz "Zaawansowane ustawienia systemu" z listy po lewej stronie.

Kliknij przycisk na dole oznaczony "Zmienne środowiskowe". Znajdź wpis `PATH` w

sekcji "Zmienne użytkownika" (*nie* "Zmienne systemowe") i kliknij "Edytuj". Jeśli nie ma wpisu PATH, kliknij "Nowy" i nazwij nową zmienną PATH.

Dodaj wszystkie trzy poniższe wpisy na końcu ścieżki *bez spacji* między nimi (ostatnie dwa wpisy pozwalają uruchomić Javę i Ant z wiersza poleceń):

```
%ANDROID_HOME%\tools
%ANDROID_HOME%\platform-tools
%JAVA_HOME%\bin
C:\Ant\bin
```

Pamiętaj, aby dołączyć separator `;`, jak pokazano (jeśli musiałeś utworzyć nową zmienną `PATH`, możesz pominąć początkowe `;`). Nie zmieniaj niczego, co już tam jest: musisz dodać to na końcu. Jeśli popełnisz błąd, kliknij "Anuluj" i spróbuj ponownie. Trzymaj się z dala od "Zmiennych systemowych". Jeśli popełnisz błąd w systemowym ustawieniu PATH, możesz spowodować poważne problemy w systemie Windows.

Będąc na tym ekranie, należy również ustawić `ANDROID_HOME` i `JAVA_HOME` zmienne środowiskowe:

1. Kliknij "OK", aby zamknąć okno dialogowe PATH, a następnie kliknij "Nowy" w sekcji "Zmienne użytkownika".
2. Ustaw nazwę na `JAVA_HOME`, a wartość na `C:\Program Files\Java\jdk1.7.0_25` (lub jakikolwiek inny katalog instalacyjny JDK).
3. Kliknij "OK", aby zamknąć okno dialogowe PATH, a następnie ponownie kliknij "New".
4. Ustaw nazwę na `ANDROID_HOME`, a wartość na `C:\Users\%USERNAME%\Android\android-sdk`.

Jeśli zainstalowałeś Javę lub Android SDK w innej lokalizacji, zmień zmienne w razie potrzeby.

Na koniec kliknij "OK", aż wszystkie okna dialogowe znikną.

Zainstaluj Node.js i npm

Node.js to platforma do tworzenia aplikacji sieciowych w JavaScript. Ma również świetny system instalowania pakietów (takich jak PhoneGap). Menedżer pakietów Node (npm) to najszybszy sposób na zainstalowanie Cordova CLI dla PhoneGap. Node będzie również używany w projektach w rozdziałach 7 i 9.

Aby zainstalować Node.js:

Windows

Pobierz i zainstaluj Node.js.

W przypadku jakichkolwiek problemów z uruchomieniem Node z wiersza poleceń w systemie Windows po jego zainstalowaniu, może być konieczne wylogowanie się i ponowne zalogowanie, aby zmiany PATH odniosły skutek.

Linux
: Zainstaluj Node.js za pomocą menedżera pakietów. Jeśli npm znajduje się w osobnym pakiecie niż Node.js, zainstaluj go również. Na przykład w systemie Ubuntu należy uruchomić polecenie `sudo apt-get install nodejs npm`.

Mac OS X
: Możesz użyć instalatora Node.js na stronie internetowej lub zainstalować go za pomocą menedżera pakietów, takiego jak Homebrew. Jeśli instalator nie doda katalogu binarnego modułu npm do PATH, może być konieczne dodanie go do PATH poprzez umieszczenie czegoś takiego w *.bash_profile* lub *.profile,* a następnie otwarcie nowego okna Terminala:

```
export PATH=/usr/local/share/npm/bin:$PATH
```

Potrzebny będzie również zainstalowany Git. Jest on domyślnie zainstalowany w systemie OS X i większości dystrybucji Linuksa. Jeśli nie, zainstaluj go za pomocą menedżera pakietów. Możesz uzyskać Git z wiersza poleceń dla systemu Windows, a instalator Git zapyta, czy chcesz zainstalować Git w zmiennej środowiskowej PATH systemu. Sugerujemy, aby to zrobić, ponieważ pozwoli to uruchomić Git z wiersza poleceń.

Instalacja Cordova CLI dla PhoneGap

Otwórz Terminal lub Wiersz polecenia i zainstaluj pakiet Cordova (daje to framework PhoneGap).

W systemie OS X lub Linux
: Uruchom następujące polecenie w wierszu Terminala:

```
$ npm install -g cordova
```

W systemie Windows
: Uruchom następujące polecenie w oknie wiersza polecenia:

```
> npm install -g cordova
```

Opcja `-g` instaluje narzędzie wiersza poleceń Cordova w lokalizacji w PATH.

Możesz teraz utworzyć projekt PhoneGap, wpisując następujące polecenie:

```
cordova create project-location nazwa-pakietu nazwa-aplikacji
```

W poleceniu `project-location` to ścieżka do nowego projektu Cordova na Androida, `package-name` to nazwa pakietu dla tworzonej aplikacji, przy użyciu notacji w stylu

odwrotnej domeny (np. com.example.myapp), a `app-name` to nazwa aplikacji.

Następnie należy przejść do katalogu projektu i dodać platformę Android:

```
cd /path/to/project-location
platforma cordova dodaj android
```

Kroki te będą się często pojawiać w kolejnych rozdziałach. W większości tej książki będziemy odnosić się do poprzednich kroków, mówiąc po prostu "użyj polecenia `cordova create`, aby utworzyć projekt".

Teraz jesteś gotowy do stworzenia swojego pierwszego projektu PhoneGap!

PhoneGap czy Cordova?

Jaka jest różnica między PhoneGap a Cordova? PhoneGap został pierwotnie opracowany przez firmę Nitobi, która została później kupiona przez Adobe. Zanim Nitobi zostało kupione, przekazało bazę kodu PhoneGap do Apache Software Foundation, aby zapewnić dobre zarządzanie kodem open source. W ramach przejścia z Nitobi do Apache, nazwa projektu została zmieniona z PhoneGap na Cordova. PhoneGap jest obecnie dystrybucją Cordova firmy Adobe.

W przeszłości PhoneGap i Cordova, projekt open source stojący za PhoneGap, były niemal identyczne. Zmieniło się to wraz z wydaniem wersji 3.0. Teraz dystrybucja Cordova zapewnia podstawową funkcjonalność do osadzania widoku internetowego (przeglądarki) w natywnej aplikacji. Cordova jest dystrybuowana bez zainstalowanych wtyczek. Pozwala to na zainstalowanie tylko potrzebnych funkcji, upraszcza bazę kodu i zmniejsza ilość kodu w aplikacji.

PhoneGap jest dystrybucją Cordova, podobnie jak Safari i Chrome są oparte na WebKit. Dystrybucja PhoneGap Cordova jest dostarczana z zainstalowanymi wszystkimi podstawowymi wtyczkami. Ponadto wiersz poleceń PhoneGap dodaje funkcje, takie jak obsługa PhoneGap Build. Jeśli nie masz zainstalowanego natywnego SDK, kompilacja projektu może zostać delegowana do PhoneGap Build.

W tej książce używamy terminów PhoneGap i Cordova zamiennie. W przypadku przykładów kodu używamy wersji Cordova o otwartym kodzie źródłowym.

Tworzenie projektu PhoneGap

Zmień katalog na dowolny katalog, w którym chcesz przechowywać projekty (być może katalog Dokumenty) i użyj polecenia `cordova create`, aby utworzyć projekt:

```
$ cordova create ~/Hello com.example.hello Hello
```

Nie ma żadnych danych wyjściowych z `create`, jeśli się powiedzie, ale utworzy nowy katalog. To polecenie utworzy katalog w katalogu roboczym o nazwie *Hello*. Zmień do niego katalogi i wyświetl listę plików:

```
$ cd ~/Hello
$ cordova platform add android
$ ls
```

Jeśli Cordova poprosi o zainstalowanie określonego celu Androida (innego niż te, które zostały zainstalowane domyślnie), wpisz `android`, aby uruchomić menedżera SDK, a następnie zainstaluj SDK, o który poprosiła Cordova. Numery wersji SDK mogą nie zawsze się zgadzać (np. Cordo- va może poprosić o Android 4.2 SDK, ale menedżer SDK oferuje tylko 4.2.2), ale cel Androida, o który prosi Cordova (np. cel 17) będzie pasował do numeru API wymienionego w menedżerze SDK, jak w Android 4.2.2 (API 17). Po zainstalowaniu celu Android, spróbuj ponownie uruchomić polecenie `cordova create`.

W systemie Windows polecenia będą nieco inne. Otwórz Wiersz poleceń (domyślnie zostaniesz umieszczony w swoim katalogu domowym, zwykle *C:\Users\username*), ale dla pewności możesz najpierw zmienić katalog na niego za pomocą `cd /D %userprofile%` (w większości konfiguracji zmienna środowiskowa `%USERPROFILE%` zawiera Twój katalog domowy). Można oczywiście uruchomić te polecenia w innym katalogu roboczym:

```
> cd /D %userprofile%
> cordova create Hello com.example.hello Hello
> cd Hello
> platforma cordova dodaj android
> reż
```

Na wszystkich trzech platformach katalog będzie zawierał następującą strukturę:

www
: Pliki zawartości projektu: obrazy, HTML itp. To z nimi będziesz pracować najczęściej.

platformy
: Obsługa plików dla różnych platform, takich jak Android.

wtyczki
: Wszelkie zainstalowane wtyczki.

fuzje
: Zasoby internetowe specyficzne dla platformy zostały połączone podczas fazy przygotowawczej.

Katalog *platforms/android* zawiera następujące elementy:

AndroidManifest.xml
: Plik manifestu opisujący strukturę aplikacji, uprawnienia itp.

ant.properties

Plik właściwości kompilatora.

aktywa
: Kopia głównego katalogu *www* projektu.

kos
z
: Wszelkie pliki binarne, które są tworzone w procesie kompilacji.

build.xml
: Szczegóły potrzebne kompilatorowi do zbudowania projektu.

cordova
: Katalog zawierający różne skrypty, których Cordova używa do automatyzacji procesu kompilacji.

gen
: Pliki wygenerowane podczas kompilacji.

libs
: Wszelkie dodatkowe wymagane biblioteki PhoneGap, takie jak biblioteka PhoneGap-NFC.

local.properties
: Właściwości środowiska programistycznego na komputerze.

proguard-project.txt
: Właściwości narzędzia do zmniejszania i zaciemniania kodu.

project.properties
: Plik właściwości projektu.

res
: Zasoby używane do tworzenia projektu. Później zmodyfikujesz niektóre pliki w

src
: tym katalogu. Pliki źródłowe Java dla aplikacji. Nie będziesz zawracać sobie nimi głowy.

W systemie Windows może być konieczne zainstalowanie sterownika USB dla urządzenia. Podczas wcześniejszej aktualizacji SDK otrzymano sterowniki USB dla wielu urządzeń. Jednak nawet flagowe urządzenia, takie jak Nexus 7, mogą wymagać oddzielnej instalacji sterownika. Na przykład można pobrać sterownik Nexus 7 ze strony ASUS. Instrukcje dotyczące instalowania sterowników dołączonych do Android SDK można znaleźć na stronie Android USB Drivers.

Masz tutaj wszystko, czego potrzebujesz, aby uruchomić bardzo podstawową aplikację, która nie robi nic poza wyświetlaniem ekranu powitalnego. Aby skompilować i zainstalować ją na swoim urządzeniu, podłącz urządzenie przez USB i uruchom następujące polecenie, jeśli korzystasz z systemu OS X lub Linux:

```
$ cordova run
```

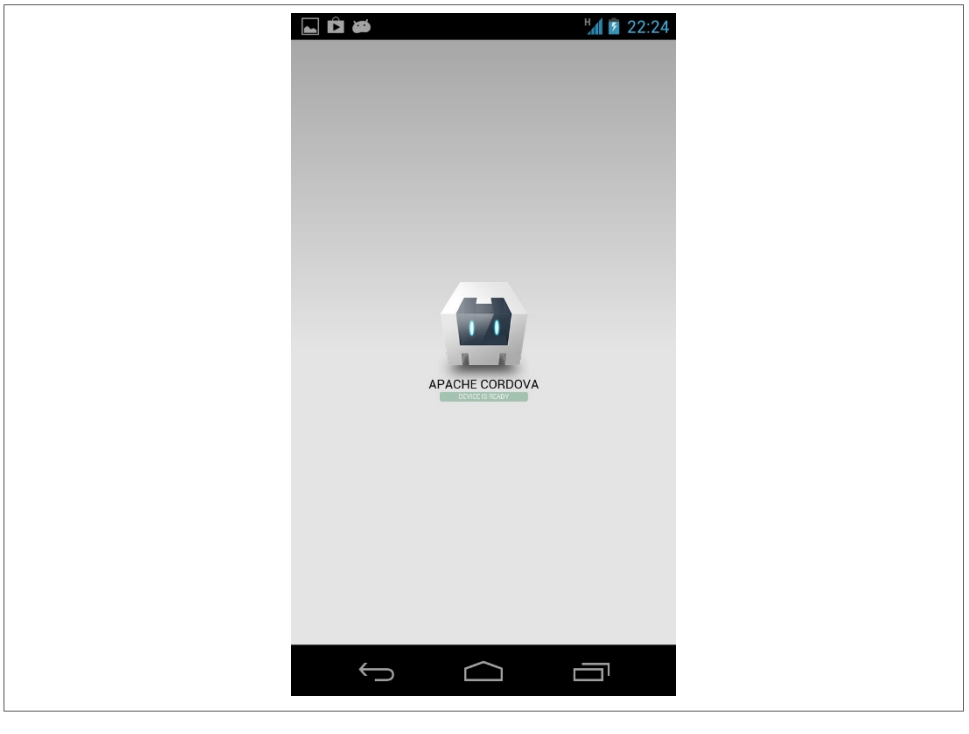
Jeśli korzystasz z systemu Windows, uruchom to polecenie:

```
> cordova run
```

Gdy wszystko będzie gotowe, aplikacja będzie działać bezpośrednio na telefonie lub tablecie z systemem Android.

Nie możesz znaleźć opcji deweloperskich na Androidzie w wersji 4.2 lub nowszej? Otwórz "Ustawienia" i przewiń w dół do "Informacje o urządzeniu" (w Nexusie 7 jest to "Informacje o tablecie"). Stuknij numer "Build" siedem razy. Spowoduje to włączenie opcji programisty. Następnie wróć do poprzedniego ekranu, aby znaleźć "Opcje deweloperskie" i włącz je. W szczególności należy włączyć "Stay Awake" podczas ładowania i "USB Debugging". Jeśli masz ustawione hasło dla swojego urządzenia, możesz je wyłączyć podczas kodowania, ponieważ eliminuje to potrzebę odblokowywania urządzenia za każdym razem, gdy przesyłasz nową wersję aplikacji.

Na koniec otwórz telefon i uruchom aplikację Hello. Powinieneś zobaczyć ekran taki jak na rysunku 3-1. Po uruchomieniu podstawowej aplikacji "Hello, World!" spróbuj zmodyfikować plik *index.html, aby uczynić* go bardziej fantazyjnym. Znajduje się on w katalogu *www*. Dodaj przyciski, obrazy i wszystkie inne rzeczy, które można dodać do strony HTML. Następnie zapisz pliki, uruchom aplikację ponownie i zobacz, jak zmienia się ona na telefonie.



Rysunek 3-1. Aplikacja PhoneGap HelloWorld

Ważne pliki

Istnieją dwa pliki, z którymi będziesz często pracować w swoich projektach PhoneGap: *in- dex.html* w katalogu *www* i plik *index.js* w katalogu *www/js.* Możesz dodać inne pliki HTML i JavaScript, ale zgodnie z konwencją będą to pliki główne aplikacji. Są to pliki, które będziesz modyfikować, aby aplikacja robiła to, co chcesz.

Strona *index.*html będzie zawierać układ graficzny aplikacji, tak jak w przypadku każdej witryny internetowej. Plik *index.js* zawiera funkcje obsługi zdarzeń, takie jak `onDeviceReady()`, `initial ize()` itd. Później dodasz do niego kilka funkcji, takich jak `onNfc()` dla odczytu tagu.

Zawsze można uzyskać niezmodyfikowaną wersję domyślnej struktury plików, tworząc nowy projekt.

Istnieje kilka innych plików JavaScript wpływających na aplikację, których prawdopodobnie nigdy nie zobaczysz: plik *cordova.*js, plik *cordova_plugins.js* i jego towarzysz, plik *cordova_plu- gins.json. Plik cordova.*js to główna biblioteka PhoneGap. Pliki *cordova_plugins.js* i *cordova_plugins.*json inicjują zainstalowane wtyczki, dając dostęp do biblioteki PhoneGap-NFC i sprzętu NFC. Pojawiają się one w *platforms/android/ assets/www po* zainstalowaniu wtyczek i uruchomieniu aplikacji.

Słowo o idiomach JavaScript

Domyślny idiom kodowania lub wzorzec programowania dla PhoneGap może wyglądać dziwnie, jeśli wciąż jesteś nowy w JavaScript. W wielu programach JavaScript zobaczysz funkcje inicjowane w ten sposób:

```
function toggleCompass() {

}
function init() {

}
```

Jeśli programowałeś w języku C lub Java, ten wzorzec prawdopodobnie wygląda znajomo. Funkcje te są globalne dla aplikacji i mogą być wywoływane z dowolnego miejsca w ten sposób:

```
toggleCompass()
```

Z drugiej strony, w domyślnym pliku Cordova *index.js widać,* że wszystkie funkcje są tworzone wewnątrz jednej gigantycznej zmiennej o nazwie `app`. W JavaScript zmienne mogą zawierać funkcje, a funkcje można przekazywać jako parametry funkcji. Ponieważ są one elementami obiektu `app`, są zapisywane jako *literał obiektu*, na przykład:

```
var app = {
   initialize: function() {
```

```
    },
```

```
        bindEvents: function() {

        },

        onDeviceReady: function() {

        }
    }
```

Literał obiektu to para nawiasów klamrowych otaczających pary nazwa/wartość, które są oddzielone przecinkami wewnątrz nawiasów klamrowych. Jeśli widziałeś *JavaScript Object Notation* (JSON), widziałeś to już wcześniej. JSON jest składnią do pisania literałów obiektów. Są to nadal funkcje, tylko w innej notacji. Aby wywołać te funkcje, które są lokalne dla obiektu `aplikacji`, należy zrobić to w następujący sposób:

```
app.initialize();
app.bindEvents();
app.onDeviceReady();
```

Pisanie wszystkich funkcji wewnątrz obiektu `aplikacji` oznacza, że wiesz, że wszystkie są lokalne i nie nadpiszesz przypadkowo funkcji z innych bibliotek JavaScript o tych samych nazwach, których możesz użyć w bardziej złożonych aplikacjach.

Więcej informacji na temat JavaScript można znaleźć w książce Douglasa Crockforda *JavaScript: The Good Parts*. Wykonuje on dobrą robotę, opisując strukturę i sposób działania JavaScript.

Prosta aplikacja lokalizatora

Ponieważ szablon "Hello, World" jest tak przydatnym odniesieniem, zachowaj go w stanie nienaruszonym i utwórz nowy projekt dla swojej pierwszej niestandardowej aplikacji. Ten projekt będzie zawierał absolutne minimum potrzebne w plikach HTML i JavaScript, aby wszystko działało. Aby było ciekawie, stworzysz aplikację, która rozpoczyna i kończy śledzenie szerokości i długości geograficznej po kliknięciu.

W poniższych przykładach, OS X i Linux (oznaczone wiodącym $) i Windows (oznaczone wiodącym >) warianty poleceń są pokazane jeden po drugim. W systemie Windows używamy %userprofile %, który w większości instalacji Windows wskazuje na katalog domowy użytkownika, ale można użyć innej lokalizacji.

W przyszłych rozdziałach, o ile polecenia nie będą się radykalnie różnić, pokażemy tylko wersję OS X i Linux, a użytkownicy Windows powinni założyć, że polecenia różnią się tak, jak pokazano tutaj.

Utwórz nowy projekt o nazwie Locator, tak jak poprzednio. Ponownie użyj `cordova create`:

```
$ cordova create ~/Locator com.example.locator Locator ❶
```

❶ Użytkownicy systemu Windows powinni wpisać `%userprofile%\Locator` zamiast `~/Locator`.

Następnie dodaj do projektu platformę Android i wtyczkę geolokalizacji. Wtyczki dla PhoneGap/Cordova rozszerzają funkcjonalność frameworka. Autorzy wtyczek rejestrują adresy URL swoich wtyczek w projekcie Cordova, aby można je było dodać do bazy danych wtyczek. Wiele wtyczek dodaje dostęp do sprzętu urządzenia, jak ta, która daje dostęp do systemu geolokalizacji urządzenia. Wtyczka NFC, którą wkrótce zobaczysz, zapewnia dostęp do radia NFC. Nieco później zobaczysz, jak wtyczki pasują do struktury plików, ale na razie oto jak zainstalować wtyczkę geolokalizacyjną:

```
$ cd ~/Locator ❶
$ cordova platform add android
$ cordova plugin add  \   ❷
    https://git-wip-us.apache.org/repos/asf/cordova-plugin-geolocation.git
```

❶ Użytkownicy systemu Windows powinni wpisać `/d %userprofile%\Locator` zamiast `~/Nfc ReaderLocator`.

❷ \ jest znakiem kontynuacji linii w Linuksie, OS X i innych systemach POSIX.

Teraz zmień katalogi na katalog *www* projektu. Edytuj plik *index.*html i usuń wszystko. Poniżej znajduje się minimalny plik HTML, który zawiera tylko podstawy, których będziesz używać w większości aplikacji. Istnieje główny element div o nazwie `app, w` którym będzie odbywać się cała akcja. Zobaczysz go i jego podelementy zmodyfikowane w pliku *index.js.* Na końcu pliku dołącz skrypt *cordova.*js i skrypt index*.js* swojej aplikacji. Na koniec wywołaj funkcję `initialize()` z pliku *index.js*, aby uruchomić aplikację:

```
<!DOCTYPE html>

<html>
   <head>
      <title>Locator</title>
   </head>
   <body style="font-size: 1.4em;">.
      <div class="app">.
         <div id="messageDiv"></div>
      </div>
      <script type="text/javascript" src="cordova.js"></script>.
      <script type="text/javascript" src="js/index.js"></script>.
      <script type="text/javascript">
       app.initialize();
     </script>
   </body>
</html>
```

Edytuj plik *index.js* i usuń wszystko. Zastąpisz go następującym skryptem. Zgodnie z konwencją, wszystkie funkcje i zmienne w tym skrypcie są lokalne dla obiektu o

nazwie app. Zacznij od zainicjowania zmiennej o nazwie app:

```
var app = {

};
```

Następnie dodaj kilka funkcji potrzebnych do inicjalizacji, aby skonfigurować nasłuchiwanie zdarzeń uruchamiania, dotknięć przycisków i tym podobnych. Zastąp dodany przed chwilą kod następującym:

```
var app = {
/*
    Konstruktor aplikacji
*/
    initialize: function() {
        this.bindEvents();
        console.log("Uruchamianie aplikacji  ❶
        Locator");
    },
/*
    powiązać wszystkie zdarzenia, które są wymagane podczas uruchamiania, ze
    słuchaczami:
*/
    bindEvents: function() {
        document.addEventListener('deviceready', this.onDeviceReady, false); ❷
    },

/*
    uruchamia się, gdy urządzenie jest gotowe do interakcji z użytkownikiem:
*/
    onDeviceReady: function() {
        app.display("Lokalizowanie...
        "); app.watchLocation();
    },
```

❶ `Konsola` jest definiowana przez przeglądarkę. Każda nowoczesna przeglądarka posiada w swoim interfejsie przeglądarkę konsoli JavaScript.

❷ jest definiowany przez przeglądarkę i wyświetlany dokument HTML.

Następnie dodaj funkcję, aby uzyskać lokalizację urządzenia. Ta funkcja ma dwa wywołania zwrotne, jedno dla sukcesu i jedno dla niepowodzenia. Jeśli się powiedzie, wyświetla współrzędne urządzenia za pomocą funkcji o nazwie `display()`. W przypadku niepowodzenia wyświetla komunikat o niepowodzeniu:

```
/*
    Wyświetla bieżącą pozycję w zakładce wiadomości:
*/
    watchLocation: function() {
        // onSuccess Callback
        // Ta metoda akceptuje obiekt `Position`, który zawiera
        // aktualne współrzędne GPS
        function onSuccess(position) {
            app.clear();
            app.display('Szerokość geograficzna: ' +
```

```
position.coords.latitude); app.display('Długość geograficzna: '
+ position.coords.longitude); app.display(new
Date().toString());
```

```
        }

        // wywołanie zwrotne onError odbiera obiekt PositionError:
        //
        function onError(error) {
            app.display(error.message);
        }

        // Opcje: zgłaszanie błędu w przypadku braku aktualizacji co 30 sekund.
        //
        var watchId = navigator.geolocation.watchPosition(onSuccess, onError, {
         timeout: 30000,
         enableHighAccuracy: true
         });
    },
```

Na koniec należy połączyć te funkcje z interfejsem użytkownika. Poniższe funkcje, `display()` i `clear()`, zapisują do elementu `div` HTML w *index.html*. Te same dwie funkcje pojawią się w wielu aplikacjach w tej książce:

```
    /*
       dołącza @message do elementu div wiadomości:
    */
    display: function(message) {
       var label = document.createTextNode(message),
          lineBreak = document.createElement("br");
       messageDiv.appendChild(lineBreak);       // dodanie podziału wiersza
       messageDiv.appendChild(label);           // dodanie tekstu
    },
    /*
       czyści div komunikatu:
    */
    clear: function() {
        messageDiv.innerHTML = "";
    }
};     // koniec aplikacji
```

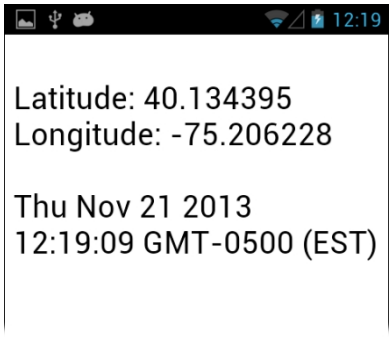
Pełną listę źródeł tej aplikacji można znaleźć w serwisie GitHub.

Zapisz edytowane pliki, a następnie `przejdź` do katalogu głównego projektu i uruchom aplikację na telefonie za pomocą następujących poleceń:

```
$ cd Locator
$ cordova run
```

```
> cd /d %userprofile%\Locator
> cordova run
```

Po uruchomieniu aplikacji na telefonie pojawi się ekran pokazany na rysunku 3-2.



Rysunek 3-2. Aplikacja Lokalizator pokazująca lokalizację

HTML i JavaScript można modyfikować tak samo, jak w przypadku każdej innej aplikacji internetowej opartej na HTML, JavaScript lub CSS. Nie krępuj się więc eksperymentować. Po zrozumieniu podstaw tworzenia aplikacji PhoneGap, możesz przejść dalej.

Sposoby debugowania

Istnieje kilka przydatnych narzędzi do debugowania podczas pracy nad aplikacjami PhoneGap. Użyj dowolnego z nich, które uznasz za najbardziej pomocne.

Po pierwsze, warto zainstalować jshint, aby sprawdzić kod przed jego uruchomieniem. Aby go zainstalować, wpisz:

```
$ npm install -g jshint
```

Następnie, aby go uruchomić, wpisz:

```
$ jshint /path/to/js/file
```

Gdy uruchomisz swój kod przez jshint, wykryje on błędy i błędy formatowania, zanim jeszcze go uruchomisz.

Gdy urządzenie jest nadal podłączone do komputera, można również uzyskać dane wyjściowe dziennika po uruchomieniu aplikacji za pomocą `adb logcat`. Po uruchomieniu aplikacji za pomocą `cordova run`, wpisz:

```
$ adb logcat
```

Daje to dziennik wszystkiego, co dzieje się na telefonie z Androidem. Jest dość szczegółowy i zawiera więcej niż tylko komunikaty *konsoli* JavaScript.*log*. Może to być przydatne, ale może też być przytłaczające. Na przykład, oto dane wyjściowe z uruchomienia poprzedniej aplikacji:

```
I/ActivityManager( 393): START u0 {act=android.intent.action.MAIN
        cat=[android.intent.category.LAUNCHER] flg=0x10200000
        cmp=com.example.locator/.Locator} z pid 590
D/dalvikvm( 2883): Opóźnione włączenie CheckJNI
```

```
I/ActivityManager( 393): Uruchom proc com.example.locator dla aktywności
```

```
        com.example.locator/.Locator: pid=2883 uid=10080
        gids={50080, 1006, 3003, 1015, 1028}

... dwadzieścia lub więcej linii wyciętych ze względu na

zwięzłość ... D/SoftKeyboardDetect( 2883): Zignoruj to

zdarzenie
I/ActivityManager( 393): Wyświetlono com.example.locator/.Locator: +475ms
D/CordovaLog( 2883): file:///android_asset/www/js/index.js: Linia 7 :
        Uruchamianie aplikacji Locator
I/Web Console( 2883): Uruchamianie aplikacji Lokalizator pod adresem
        file:///android_asset/ www/js/index.js:7
```

Jeśli chcesz odfiltrować wszystkie komunikaty oprócz tych z konsoli internetowej, przeprowadź je przez `grep` (OS X lub Linux) lub `findstr` (Windows) w następujący sposób:

```
$ adb logcat | grep "Web Console"
```

```
> adb logcat | findstr /C: "Web Console"
```

Gdy to zrobisz, wszystkie instrukcje `console.log()` w twoim JavaScripcie pojawią się w konsoli. Skupi się na rzeczach, które musisz wiedzieć przez większość czasu. Dane wyjściowe są ograniczone do:

```
I/Web Console( 2335): Uruchamianie aplikacji Lokalizator
        at file:///android_asset/www/js/index.js:7
```

W ten sposób można również filtrować według dowolnego innego typu wiadomości. Aby zamknąć program, naciśnij klawisze ctrl+C. Po odłączeniu urządzenia od komputera aplikacja wyłączy się automatycznie.

Niektórzy programiści wolą utrzymywać ją uruchomioną w oddzielnym oknie Terminala lub Wiersza poleceń, aby mogli ją skompilować i uruchomić bez konieczności jej uruchamiania i zatrzymywania. Można rozróżnić między uruchomieniami aplikacji, naciskając Enter kilka razy w tym oknie, aby utworzyć przerwę. Inni wolą metodę stop/start, ponieważ zapewnia ona wyraźniejszy zapis za każdym razem, gdy testujesz swoją aplikację.

Możesz także użyć Android Debug Monitor, aby obserwować wszystko, co się dzieje. Jest to graficzny interfejs użytkownika, który znajduje się w podkatalogu *narzędzi* Android SDK. Ponieważ dodałeś go wcześniej do swojej ścieżki, możesz po prostu wpisać następujące polecenie, aby go uruchomić:

Linux lub OS X:

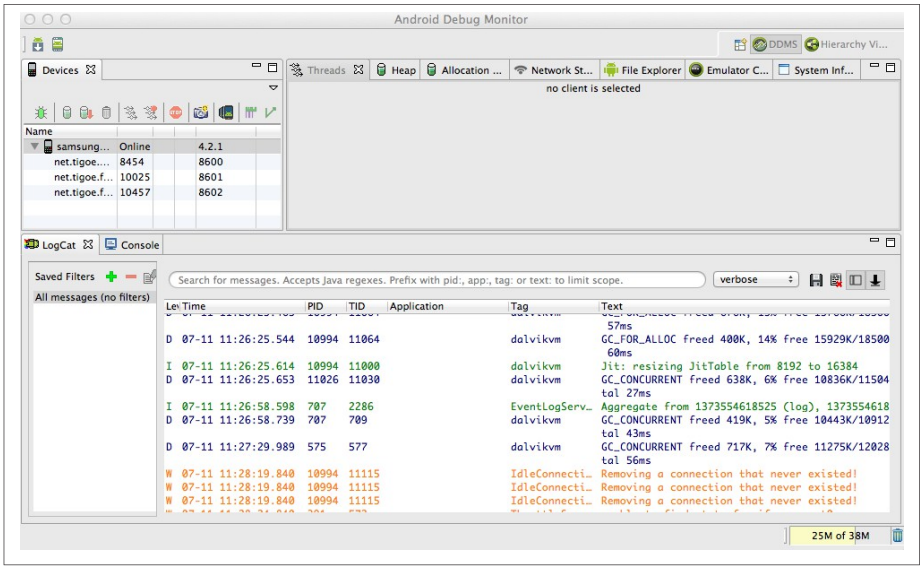
```
$ monitor &
```

Windows:

```
> uruchom monitor
```

Ampersand spowoduje uruchomienie monitora w tle i powrót do Wiersza poleceń po uruchomieniu. Jeśli korzystasz z systemu Windows, plik wsadowy uruchamia

monitor w tle. Komunikaty logcat można filtrować za pomocą tagu "Web Console" lub dowolnego innego tagu dziennika, którego chcesz szukać. Możesz także zobaczyć status innych dzienników. Dostępny jest również monitor wydajności, eksplorator plików i wiele innych. Rysunek 3-3 przedstawia interfejs.



Rysunek 3-3. Monitor debugowania systemu Android

Jeśli urządzenie jest podłączone do Internetu, konsolę JavaScript można również znaleźć na stronie http://debug.phonegap.com. Wystarczy wpisać unikalną nazwę, taką jak "kramnitz", we wpisie pola na tej stronie, a następnie umieścić łącze do skryptu w pliku *index.html*. Następnym razem, gdy uruchomisz swoją aplikację, zobaczysz konsolę JavaScript pod adresem *http://debug.phonegap.com/client/#kramnitz*. Jest wygodna i znajoma, ale wymaga połączenia z siecią.

Zaletą *debug.phonegap*.com jest również jego wada: widzisz tam tylko błędy skryptów Java. Czasami wyłapujesz błędy z komunikatów Androida za pomocą `adb log cat`, które w przeciwnym razie przegapiłbyś, gdybyś szukał tylko błędów JavaScript. Może się również okazać, że *debug.phonegap.com* działa nieco wolniej, gdy jest na nim duże obciążenie innych programistów.

PhoneGap spotyka NFC: czytnik NFC

Teraz, gdy masz już do czynienia z PhoneGap, nadszedł czas, aby rozszerzyć go o wtyczkę NFC. W tym celu potrzebne będzie to samo oprogramowanie, co w przypadku pierwszej aplikacji, wtyczka PhoneGap-NFC oraz kilka tagów NFC. Pierwsza utworzona aplikacja NFC odczyta tylko unikalny identyfikator każdego tagu, który zobaczy.

Ta sama struktura będzie używana dla wszystkich aplikacji PhoneGap, więc zachowaj tę aplikację pod ręką, aby użyć jej jako szablonu. Utwórz nową aplikację o nazwie `NfcReader`:

```
$ cordova create ~/NfcReader com.example.nfcreader NfcReader ❶
$ cd ~/NfcReader ❷
$ cordova platform add android
```

❶ Użytkownicy systemu Windows powinni wpisać `%userprofile%\NfcReader` zamiast `~/NfcReader`. ❷ Użytkownicy systemu Windows powinni wpisać `/d %userprofile%\NfcReader` zamiast `~/NfcReader`.

Instalacja wtyczki NFC

Istnieje szereg zmian potrzebnych do dodania obsługi NFC do projektu. Wszystkie one są zautomatyzowane przez polecenie:

```
$ cordova plugin add /path/to/plugin
```

Ten, który będzie często używany w całej książce, to:

```
$ cordova plugin add https://github.com/chariotsolutions/phonegap-nfc.git
```

Aby zrozumieć, co dzieje się pod maską, oto co jest zainstalowane:

- Pliki wtyczki znajdują się w podkatalogu *plugins/* projektu (w tym przypadku jest to *com.chariotsolutions.nfc.plugin*). Zostaje ona również zainstalowana w *platformach/android/zasoby/www/wtyczki/*.
- Plik *AndroidManifest.xml* projektu i pliki konfiguracyjne zasobów również wymagają modyfikacji, aby aplikacja mogła korzystać ze sprzętu czytnika NFC na urządzeniu. Plik ten znajduje się w katalogu *platforms/android/ projektu.*
- Plik *platforms/android/res/xml/config.xml* również zostaje zmodyfikowany. Oto modyfikacje tych dwóch plików:

```
<feature name="NfcPlugin">.
   <param name="android-package"
    value="com.chariotsolutions.nfc.plugin.NfcPlugin" />
</funkcja>
```

- Plik *AndroidManifest.xml* otrzymuje uprawnienie NFC dodane na końcu innych znaczników uprawnień:

```
<uses-permission android:name="android.permission.NFC" />
<uses-feature android:name="android.hardware.nfc"
              android:required="false" />
```

Ponieważ jest to dużo do zapamiętania, miło jest, że jednowierszowe polecenie `dodawania` wtyczki robi to wszystko za Ciebie.

Utworzyłeś już swój projekt za pomocą polecenia `create`, zgodnie z wcześniejszymi instrukcjami. Następnie zmień katalogi na katalog główny projektu i dodaj wtyczkę w sposób opisany na początku tej sekcji:

```
$ cd ~/NfcReader
$ cordova plugin add https://github.com/chariotsolutions/phonegap-nfc.git
```

```
> cd /d %userprofile%\NfcReader
> cordova plugin add https://github.com/chariotsolutions/phonegap-nfc.git
```

Spowoduje to zainstalowanie wtyczki NFC z repozytorium online w bieżącym katalogu roboczym.

Pisanie aplikacji czytnika NFC

Teraz, gdy masz już wszystkie zasoby na miejscu, nadszedł czas, aby stworzyć samą aplikację. W tym projekcie będziesz musiał edytować dwa pliki:

- *index.html* w katalogu *www*
- *index.js* w katalogu *www/js*

Aplikacja ta będzie nasłuchiwać tagów RFID kompatybilnych z NFC i drukować ich identyfikatory na ekranie.

Zacznij od *index.html,* jak poprzednio. Oto strona z gołym kodem, podobna do strony indeksu aplikacji Locator. Zmiany polegają na dodaniu wywołania skryptu Cordova na końcu treści i zmianie elementów div aplikacji w treści. Ten ostatni jest miejscem, w którym będą wyświetlane identyfikatory tagów.

Otwórz plik *index.html* i zastąp go poniższym kodem, a następnie zapisz zmiany:

```
<!DOCTYPE html>

<html>
   <head>
      <title>Czytnik identyfikatorów tagów NFC</title>
   </head>
   <body style="font-size: 1.5em">.
      <div class="app">.
         <div id="messageDiv"></div>
      </div>
      <script type="text/javascript" src="cordova.js"></script>.
      <script type="text/javascript" src="js/index.js"></script>.
      <script type="text/javascript">
       app.initialize();
     </script>
   </body>
</html>
```

Plik *index.js* będzie miał program obsługi zdarzeń, który będzie nasłuchiwał tagów NFC i przepisywał div wiadomości, gdy otrzyma tag. Tak jak poprzednio, zacznij od zainicjowania zmiennej app i dodania funkcji `initialize()` oraz funkcji `bindEvents()`:

```
var app = {
/*
```

```
    Konstruktor aplikacji
*/
```

```
    initialize: function() {
        this.bindEvents();
        console.log("Uruchamianie aplikacji NFC
        Reader");
    },

/*
    powiązać wszelkie zdarzenia, które są wymagane podczas uruchamiania, ze
    słuchaczami:
*/
    bindEvents: function() {
        document.addEventListener('deviceready', this.onDeviceReady, false);
    },
```

Funkcja `onDeviceReady()` jest tym razem nieco bardziej złożona. Konieczne jest dodanie słuchacza, gdy tagi zostaną wykryte przez czytnik NFC. Funkcja `nfc.addTagDiscovered Listener()` informuje wtyczkę NFC, aby powiadamiała aplikację o odczytaniu dowolnego tagu NFC. Pierwszym argumentem jest procedura obsługi zdarzenia, która jest wywoływana po zeskanowaniu tagu. Drugi i trzeci argument to wywołania zwrotne powodzenia i niepowodzenia inicjalizacji wtyczki. Jeśli się powiedzie, zostanie wywołana funkcja `onNfc()`. Gdy słuchacz zostanie zainicjowany, poinformuje o tym użytkownika, a gdy się nie powiedzie, przekaże również komunikaty o niepowodzeniu.

Funkcja obsługi `addTagDiscoveredListener()` jest jednym z kilku różnych detektorów, których można używać w bibliotece NFC. Jest najbardziej ogólny. Nie dba o to, co znajduje się na tagu, po prostu reaguje za każdym razem, gdy widzi kompatybilny tag:

```
/*
    uruchamia się, gdy urządzenie jest gotowe do interakcji z użytkownikiem:
*/
    onDeviceReady: function() {

        nfc.addTagDiscoveredListener(
            app.onNfc,              // tag został pomyślnie zeskanowany
            function (status) {     // słuchacz został pomyślnie zainicjowany
                app.display("Dotknij tagu, aby odczytać jego identyfikator.");
            },
            function (error) {      // słuchacz nie został zainicjowany
                app.display("Nie udało się zainicjować czytnika NFC " +
                    JSON.stringify(error));
            }
        );
    },
```

Funkcja `onNfc()` pobiera tag odczytany w zdarzeniu NFC i drukuje go na ekranie za pomocą funkcji `display()`, podobnie jak w aplikacji Locator. Są to ostatnie funkcje w aplikacji:

```
/*
    wyświetla identyfikator tagu z @nfcEvent w divie wiadomości:
```

```
*/
    onNfc: function(nfcEvent) {
        var tag = nfcEvent.tag;
        app.display("Read tag: " + nfc.bytesToHexString(tag.id));
```

```
    },

/*
    dołącza @message do elementu div wiadomości:
*/
    display: function(message) {
        var label = document.createTextNode(message),
            lineBreak = document.createElement("br");
        messageDiv.appendChild(lineBreak);       // dodanie podziału wiersza
        messageDiv.appendChild(label);           // dodaj tekst
    },
/*
    czyści div komunikatu:
*/
    clear: function() {
        messageDiv.innerHTML = "";
    }
};      // koniec aplikacji
```

Pełną listę źródeł tej aplikacji można znaleźć w serwisie GitHub.

To wszystkie zmiany. Zapisz oba pliki, a następnie zmień katalogi na katalog główny projektu, jeśli jeszcze tam nie jesteś, a następnie skompiluj i zainstaluj aplikację:

```
$ cd NfcReader
$ cordova run
```

```
> cd /d %userprofile%\NfcReader
> cordova run
```

Spowoduje to uruchomienie aplikacji NFC Reader. Po jej uruchomieniu zbliż tag do telefonu, a powinien pojawić się komunikat "Odczytaj tag", a następnie identyfikator UID w zapisie szesnastkowym, jak pokazano na rysunku 3-4. Spróbuj z innym tagiem, a zobaczysz inny identyfikator UID.

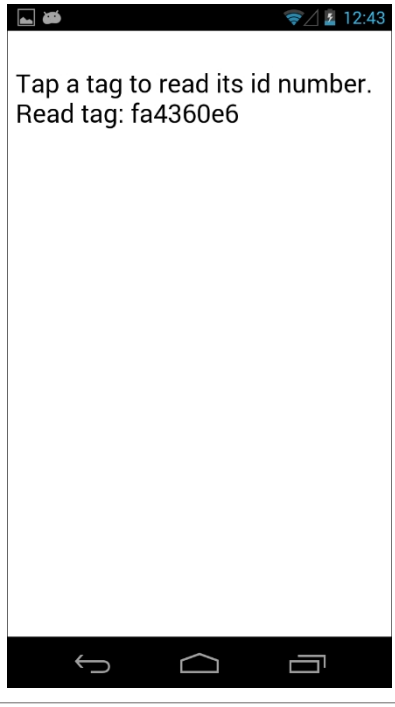
Zarządzanie repozytorium projektu

Po uruchomieniu projektu, jeśli zamierzasz zapisać go w repozytorium Git lub innym systemie kontroli wersji, możesz nie chcieć przechowywać plików kompilacji, plików binarnych itp. Dobrym pomysłem jest usunięcie lub `.gitignore` pliku *platforms/android/local.proper- ties* w katalogu projektu, aby nie został przesłany do repozytorium. Jeśli zmienne PATH zostały ustawione w profilu logowania, jak pokazano w sekcji "Instalowanie Node.js i npm" na stronie 27, plik *local*.properties nigdy nie będzie potrzebny. W przeciwnym razie, podczas aktualizacji projektu ze zdalnego repozytorium, będzie on potrzebny:

```
$ cd projectDirectory
$ android update project -p platforms/android/
```

Oto lista wszystkich elementów, które można bezpiecznie pominąć w repozytorium. Jeśli korzystasz z Git, możesz je po prostu `.gitignore`:

```
platform/android/local.properties
platform/android/bin platform/android/gen
platform/android/assets/www
```



Rysunek 3-4. Aplikacja NFC Reader odczytująca tag

Będzie to działać z każdym tagiem RFID, który jest kompatybilny z czytnikiem NFC. Obejmuje to wszystkie tagi zgodne z formatem ISO-14443A, w tym tagi Philips i NXP Mifare, tagi Sony FeliCa, NXP DESFire. Nie odczytujesz jeszcze ani nie zapisujesz danych na tagu; odczytujesz tylko unikalny identyfikator tagu. W przypadku tej aplikacji kompatybilne tagi nie są tak dużym problemem, jak w przypadku rozpoczęcia zapisywania rekordów NDEF do tagów. Przekonasz się, że ta aplikacja może odczytywać identyfikatory tagów Mifare Classic nawet na urządzeniach, które są rzekomo niekompatybilne z Mifare Classic.

Udowodniłeś, że czytnik twojego urządzenia działa i można go zaprogramować do odczytu tagów. W następnym rozdziale dowiesz się, jak odczytywać i zapisywać tagi przy użyciu formatu wymiany danych NFC, NDEF.

Rozwiązywanie problemów

Jeśli aplikacja nie odczytuje żadnego z tagów, oto kilka rzeczy do sprawdzenia:

Czy urządzenie obsługuje technologię NFC?
: Aby to sprawdzić, przejdź do "Ustawień" na swoim urządzeniu i dotknij "Więcej...". Jeśli nie widzisz tam NFC, nie masz szczęścia. Jeśli ją widzisz, upewnij się, że jest włączona. Po zeskanowaniu tagu powinna pojawić się melodia potwierdzenia. Melodia udanego odczytu kończy się wysokim dźwiękiem, a nieudany odczyt kończy się niskim dźwiękiem. Wtyczka PhoneGap-NFC zgłosi błąd, jeśli NFC jest wyłączone lub jeśli NFC nie jest dostępne. Ta aplikacja drukuje te informacje na ekranie. Inne mogą tego nie robić.

Czy używasz kompatybilnych tagów?
: Jeśli po zeskanowaniu tagu nie otrzymasz melodii potwierdzenia, upewnij się, że jest to kompatybilny tag. Jeśli nie, czytnik go nie odczyta.

Czy nawiązałeś dobry kontakt?
: Czytnik NFC w większości urządzeń znajduje się tylko z tyłu urządzenia, zwykle w pobliżu górnej części. Jeśli nie uzyskasz odczytu, spróbuj przesunąć tag z tyłu, aby dowiedzieć się, gdzie znajduje się antena czytnika. Czasami uzyskanie dobrego odczytu zajmuje około sekundy.

Czy identyfikatory są uszkodzone?
: Podarte, pomarszczone znaczniki nie będą działać, podobnie jak znaczniki umieszczone prostopadle do powierzchni urządzenia. Upewnij się, że znaczniki są w dobrym stanie i przylegają płasko do urządzenia.

Wnioski

Kroki, które wykonałeś w tym rozdziale, będą używane we wszystkich projektach PhoneGap-NFC, które pojawią się w tej książce, więc oto krótkie podsumowanie do wykorzystania w przyszłości:

- Zainicjuj ogólną aplikację PhoneGap za pomocą polecenia `cordova create`
- Dodaj platformę za pomocą `platformy cordova`
- Dodaj zasoby PhoneGap-NFC za pomocą `wtyczki cordova`

Aplikacja wymaga co najmniej następujących plików:

- *index.html*, w katalogu *www*
- *index.js*, w katalogu *www/js*

Plik *index.html* musi zawierać następujące skrypty JavaScript:

- *cordova.js*

- *index.js*

Główny plik JavaScript powinien zawierać następujące elementy z biblioteki PhoneGap- NFC:

- Obsługa listenera NFC
- Funkcja reagująca na zdarzenia NFC

Jeśli masz wszystkie te elementy, jesteś gotowy do pracy.

Do tej pory powinieneś być zaznajomiony ze strukturą aplikacji PhoneGap i krokami niezbędnymi do dodania wtyczki PhoneGap-NFC do aplikacji. Zobaczysz tę samą strukturę powieloną w pozostałych projektach w tej książce. Jednak to, co do tej pory widziałeś, to tak naprawdę tylko RFID - nie było jeszcze prawdziwej akcji NFC. W następnym rozdziale dowiesz się o formacie wymiany danych, który definiuje wszystkie transakcje NFC.

ROZDZIAŁ 4

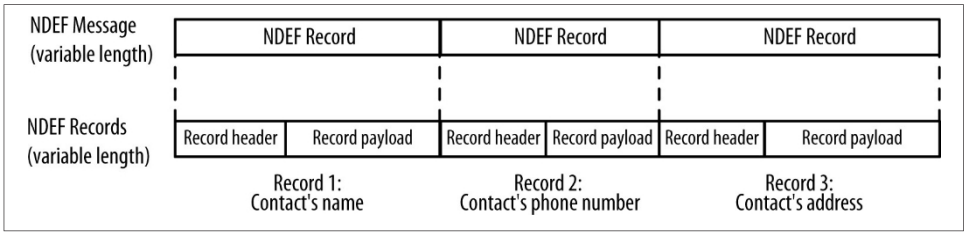
Przedstawiamy NDEF

Aby zrozumieć NFC, musisz wiedzieć o formacie wymiany danych NFC (NDEF), który jest językiem franca dla urządzeń i tagów NFC. W tym rozdziale dowiesz się o strukturze NDEF i zawartych w nim rekordach. Napiszesz także kilka aplikacji, które odczytują i zapisują wiadomości w formacie NDEF.

Struktura NDEF

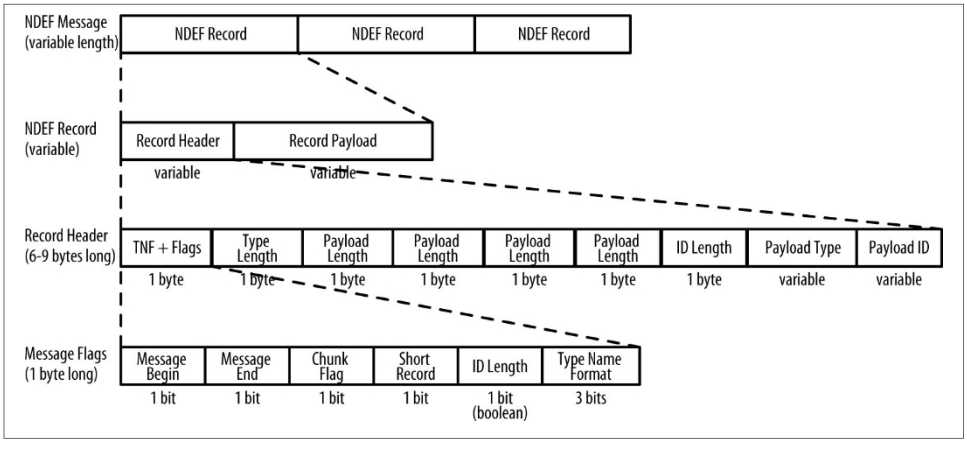
NDEF jest formatem binarnym o strukturze *wiadomości*, z których każda może zawierać kilka *rekordów*, jak pokazano na rysunku 4-1. Każdy rekord składa się z *nagłówka,* który zawiera metadane dotyczące rekordu, takie jak typ rekordu, długość i tak dalej, oraz *ładunku*, który zawiera treść wiadomości. Pomyśl o wiadomości NDEF jak o akapicie, a o rekordach jak o zdaniach w nim zawartych. Dobrze sformułowany akapit składa się ze zdań odnoszących się do jednego tematu. Podobnie, dobrą praktyką jest używanie jednego komunikatu NDEF składającego się z kilku rekordów do opisania jednego tematu, na przykład wpisu w książce adresowej.

Transakcje NFC są zazwyczaj krótkie. Każda wymiana zazwyczaj składa się tylko z jednej wiadomości, a każdy tag przenosi tylko jedną wiadomość. Należy pamiętać o fizycznych okolicznościach wymiany NFC: dotykasz swojego urządzenia do innego urządzenia lub tagu, a cała wymiana odbywa się, gdy jesteś w kontakcie z drugim urządzeniem lub tagiem. Nie chcesz wysyłać całej powieści w jednej wymianie, więc pomyśl o swoich wiadomościach NDEF jako o długości akapitu, a nie książki. W jednym z ostatnich rozdziałów tej książki znajdziesz obejście tego problemu w przypadku wysyłania dużych plików, ale na razie traktuj jedną wymianę NFC jako jedną wiadomość NDEF i myśl o jednej wiadomości NDEF jako o jednym lub kilku krótkich rekordach.



Rysunek 4-1. Struktura komunikatu NDEF składającego się z kilku rekordów; jest to typowy przykład - wpis książki adresowej z trzema rekordami (nazwa, numer telefonu, adres).

Rekord NDEF zawiera ładunek danych i metadane opisujące sposób interpretacji ładunku. Ładunek każdego rekordu może być jednym z kilku różnych *typów danych*. Nagłówek każdego rekordu zawiera metadane opisujące rekord i jego miejsce w wiadomości, a następnie jego typ i identyfikator. Po nagłówku następuje ładunek. Rysunek 4-2 przedstawia pełny obraz bitów i bajtów rekordu NDEF.



Rysunek 4-2. Struktura wiadomości NDEF, ze szczegółami dotyczącymi bajtów nagłówka

Jak widać na rysunku 4-2, rekord NDEF składa się z formatu nazwy typu (TNF), typu ładunku, identyfikatora ładunku i ładunku. Ładunek jest najważniejszą częścią rekordu NDEF; jest to zawartość, którą przesyłasz. TNF mówi, jak interpretować typ ładunku. Typ ładunku jest typem specyficznym dla NFC, typem nośnika MIME lub identyfikatorem URI, który mówi, jak interpretować ładunek. Innym sposobem myślenia o tym jest to, że TNF to metadane dotyczące typu ładunku, a typ ładunku to metadane dotyczące ładunku. Identyfikator ładunku jest opcjonalny i umożliwia powiązanie lub wzajemne odniesienie wielu ładunków.

Nazwa typu Format

Rekordy NDEF rozpoczynają się od formatu nazwy typu. TNF wskazuje strukturę wartości pola typu. TNF mówi, jak interpretować pole typu. Istnieje siedem możliwych wartości TNF:

0 Pusty
: Pusty rekord, który nie ma typu ani ładunku.

1 Dobrze znany
: Jeden z kilku predefiniowanych typów określonych w specyfikacji NFC Forum RTD.

2 Typ nośnika MIME
: Typ nośnika internetowego zdefiniowany w dokumencie RFC 2046.

3 Bezwzględny URI
: Identyfikator URI zdefiniowany w dokumencie RFC 3986.

4 Zewnętrzne
: Wartość zdefiniowana przez użytkownika w oparciu o zasady zawarte w specyfikacji definicji typu rekordu forum NFC.

5 Nieznany
: Typ jest nieznany. Długość typu musi wynosić 0.

6 Bez zmian
: Tylko dla środkowych i końcowych rekordów pakietów. Długość typu musi wynosić 0.

7 Zarezerwowane
: Zarezerwowany przez NFC Forum do wykorzystania w przyszłości.

W przypadku wielu aplikacji prawdopodobnie użyjesz TNF 01 (Well-Known) lub TNF 02 (MIME media-type) dla różnych mediów internetowych. Często zobaczysz również TNF 04 (External), ponieważ Android używa typu zewnętrznego o nazwie Android Application Record do uruchamiania aplikacji.

Typ ładunku

Typ Payload, znany również jako typ Record, opisuje bardziej szczegółowo zawartość ładunku. TNF definiuje format typu ładunku. Typ może być typem określonym przez NDEF, typem MIME, URI lub typem zewnętrznym. Specyfikacja NDEF Record Type Def- inition (RTD) opisuje znane typy rekordów i określa zasady tworzenia typów zewnętrznych. MIME RFC i URI RFC określają zasady dla innych typów.

Na przykład rekord TNF 01 (Well-Known) może mieć typ rekordu "T" dla wiadomości tekstowej, "U" dla wiadomości URI lub "Sp", jeśli ładunek jest inteligentnym plakatem. Rekord

TNF 02 (MIME media-type) może być jednym z kilku różnych typów rekordów, między innymi "text/html", "text/json" i "image/gif". Rekord TNF 03 (Absolute URI) miałby dosłowny URI, http://schemas.xmlsoap.org/soap/envelope/, jako typ. Rekord TNF 04 (External) może również mieć jeden z kilku różnych typów rekordów, z których najczęstszym jest "android.com:pkg".

Więcej szczegółów można znaleźć w dokumencie specyfikacji NDEF dotyczącym formatów definicji typów rekordów forum NFC. Listę popularnych specyfikacji NFC RTD można również znaleźć w Załączniku A.

Wiadomości NDEF mogą zawierać wiele typów Payload, ale zgodnie z konwencją typ pierwszego rekordu określa sposób przetwarzania całej wiadomości. Na przykład, filtrowanie intencji Androida dla wiadomości NDEF analizuje tylko pierwszy rekord.

URI w komunikatach NDEF

Terminy URI, URL i URN będą często używane w tej książce. URI, czyli jednolity identyfikator zasobu, to ciąg znaków identyfikujący zasób sieciowy. URN, czyli jednolita nazwa zasobu, określa URI. URL, czyli jednolity lokalizator zasobów, informuje również o typie protokołu transportowego potrzebnego do uzyskania zasobu. Jeśli URN jest twoim imieniem i nazwiskiem, to URI jest twoim adresem, a URL mówi komuś, aby wsiadł do autobusu, aby dostać się pod twój adres. Lub w terminologii internetowej:

URN
: mySpecialApp

URI
: net.tigoe.mySpecialApp (w formacie odwróconej domeny)

URL
: *http://tigoe.net/mySpecialApp*

Format nazwy typu "Absolute URI" jest nieco mylący. Bezwzględny URI TNF oznacza, że *typ rekordu*, a nie *ładunek*, jest URI. URI w polu typu opisuje ładunek, podobnie jak typ MIME opisuje ładunek dla TNF 02. Na przykład: Windows i Windows Phone używają TNF 03 (bezwzględny URI) dla rekordów LaunchApp przy użyciu URI "windows.com/LaunchApp". Rekordy LaunchApp monitują użytkownika o uruchomienie aplikacji, podobnie jak Android używa rekordów aplikacji Androida do uruchamiania aplikacji. Jeśli aplikacja nie jest zainstalowana, użytkownik jest proszony o pobranie jej ze sklepu.

Android łamie nieco specyfikację NDEF z rekordami Absolute URI. Chociaż specyfikacja NDEF mówi, że typ opisuje ładunek, urządzenia z Androidem obsługują rekordy TNF 03 (Absolute URI), otwierając przeglądarkę z URI w polu typu; basically, Android traktuje URI w polu typu tak, jakby był ładunkiem. BlackBerry i Windows Phone nie otwierają przeglądarki, gdy skanowany jest znacznik Absolute URI.

Jeśli chcesz wysłać URI lub URL jako *ładunek*, nie powinieneś używać TNF 03 (Absolute URI). Należy zakodować je jako TNF 01 (Well-Known) z NFC RTD "U" (URI). Specyfikacja NDEF zapewnia specyfikację definicji typu rekordu URI z kodami identyfikatorów URI w celu wydajniejszego kodowania identyfikatorów URI. Na przykład 0x01 jest kodem dla http://www. 0x02 jest kodem dla https://www. W sekcji "Pisanie różnych typów rekordów" na stronie 66 zobaczysz przykład, w którym dodajesz adres URL do ładunku, a następnie dodajesz pojedynczy bajt o wartości 0x01, aby dodać http://www. Zobaczysz je bardziej szczegółowo w poniższej aplikacji, a wszystkie możesz zobaczyć w Załączniku A.

Identyfikatory URI można również zakodować w TNF 01 (Well-Known) z NFC RTD "Sp" (Smart Poster). Korzystanie z rekordów Smart Poster umożliwia dołączenie do identyfikatora URI dodatkowych informacji, takich jak opisy tekstowe w wielu językach, ikony i opcjonalne instrukcje przetwarzania.

Identyfikator ładunku

Identyfikator ładunku, który jest polem opcjonalnym, powinien być prawidłowym identyfikatorem URI. Może to być względny URI, więc nawet "foo" będzie działać. Służy do umożliwienia aplikacjom identyfikowania ładunku w rekordzie za pomocą identyfikatora lub umożliwienia innym ładunkom odwoływania się do innych ładunków. To ty decydujesz o identyfikatorze ładunku, ale nie musisz go dołączać.

Ładunek

Ładunek to zawartość. Może to być cokolwiek, co mieści się w strumieniu bajtów. Prawidłowo skonstruowana biblioteka NDEF nie dba o to, co znajduje się w ładunku, po prostu przekazuje go dalej. Możesz zaszyfrować swój ładunek, możesz wysłać zwykły tekst, możesz wysłać binarnego bloba lub cokolwiek innego, co możesz wymyślić. Do aplikacji wysyłającej i odbierającej należy uzgodnienie, co oznacza ładunek i jak jest sformatowany.

Rysunek 4-2 zawiera więcej szczegółów na temat formatu binarnego pierwszego bajtu nagłówka rekordu. Miejmy nadzieję, że nigdy nie będziesz musiał korzystać z tych informacji, ponieważ każda dobrze napisana biblioteka NDEF powinna sobie z tym poradzić. Jeśli korzystasz z bibliotek programistycznych opisanych w tej książce, to prawda. Szczegółami formatowania binarnego zajmiemy się za ciebie i możesz bezpiecznie przejść do "NDEF w praktyce" na stronie 56. Jeśli chcesz lepiej zrozumieć strukturę komunikatów i rekordów, czytaj dalej.

Układ zapisu

Pierwsze pięć bitów rekordu NDEF to flagi określające sposób przetwarzania rekordu i informacje o jego lokalizacji w wiadomości.

Flagi bitowe w pierwszym bajcie nagłówka rekordu są następujące:

MB (Początek wiadomości)

Prawda, gdy jest to pierwszy rekord w wiadomości.

ME (koniec wiadomości)
: Prawda, gdy jest to ostatni rekord w wiadomości.

CF (Chunk Flag)
: Prawda, gdy rekord jest podzielony na fragmenty.

SR (Short Record)
: Prawda, jeśli używany jest format krótkiego rekordu dla długości ładunku.

IL (długość ID jest obecna)
: Prawda, jeśli pole ID Length jest obecne.

Należy pamiętać, że bit IL nie jest długością pola ID, a jedynie wskazuje obecność pola długości. Jeśli bit IL ma wartość 0, nie ma pola długości ID ani pola ID.

Flagi bitowe Message Begin i Message End są używane do przetwarzania rekordów w komunikacie. Ponieważ komunikat NDEF jest tylko zbiorem jednego lub więcej rekordów NDEF, nie ma formatu binarnego dla komunikatu NDEF. Flagi MB i ME pozwalają określić moment rozpoczęcia i zakończenia wiadomości. Pierwszy rekord w wiadomości będzie miał flagę MB ustawioną na true. Środkowe rekordy będą miały obie flagi ustawione na false. Końcowy rekord w wiadomości będzie miał flagę ME ustawioną na true. Wiadomość z jednym rekordem będzie miała oba bity Message Begin i Message End ustawione na true.

Ponieważ istnieje tylko osiem możliwych formatów nazw typów, do ich przechowywania potrzebne są tylko 3 bity. TNF jest przechowywany w ostatnich trzech bitach bajtu Message Flags.

Nagłówek rekordu

Rekordy NDEF są strukturami danych o zmiennej długości. Nagłówek rekordu zawiera informacje wymagane do odczytu danych.

Rekord NDEF zaczyna się od bajtu TNF, który zawiera flagi bitowe. Po TNF nagłówek rekordu NDEF zawiera długość typu. Długość typu to jednobajtowe pole określające długość typu ładunku w bajtach. Długość typu jest wymagana, ale może wynosić zero.

Długość ładunku jest następna. Flaga bitowa Short Record (SR) w pierwszym bajcie nagłówka rekordu określa długość rekordu payload. Jeśli SR ma wartość true, długość ładunku wynosi jeden bajt, w przeciwnym razie cztery bajty. Długość ładunku jest wymagana, ale może wynosić zero.

Jeśli pole długości ID jest obecne, a flaga (IL) ma wartość true, następnym bajtem w nagłówku jest długość ID.

Pole typu rekordu jest polem o zmiennej długości po polu długości ID (lub po polu długości ładunku, jeśli flaga IL jest fałszywa). Pole długości typu określa, ile bajtów należy odczytać.

Jeśli istnieje identyfikator rekordu, pojawia się on po typie. To pole o zmiennej długości jest określane przez bajt długości ID.

To jest koniec nagłówka. Następny jest ładunek.

Jak duża może być wiadomość NDEF?

Ładunek rekordów NDEF jest ograniczony do 232-1 bajtów, dlatego pole długości ładunku w nagłówku wynosi cztery bajty (lub ^{232} bity). Rekordy mogą być jednak łączone w wiadomości w łańcuchy, tworząc dłuższe ładunki. Teoretycznie nie ma ograniczeń co do długości wiadomości NDEF. W praktyce, możliwości urządzeń i tagów określają limity. Jeśli wymieniasz wiadomości peer-to-peer między urządzeniami i nie są zaangażowane żadne tagi, twoje wiadomości NDEF są ograniczone tylko przez pojemność obliczeniową twoich urządzeń i cierpliwość osoby trzymającej dwa urządzenia razem. Jeśli jednak komunikujesz się między urządzeniem a tagiem, twoje wiadomości są ograniczone pojemnością pamięci tagu.

W przypadku korzystania z tagów NFC limity rozmiaru rekordów są znacznie poniżej limitu $^{232-1}$ bajtów. Typy tagów NFC są oparte na kilku różnych standardach tagów RFID. Większość typów tagów NFC opiera się na standardzie ISO-14443A. Różnią się one od 96 bajtów pamięci, z możliwością rozszerzenia do 4K w zależności od typu tagu. Rodzina tagów Philips/NXP Mifare jest kompatybilna z NFC, w tym Mifare Ultralights, Mifare Classic 1K i 4K oraz Classic Mini. Istnieje jeden typ tagu NFC oparty na japońskim standardzie przemysłowym (JIS) X 6319-4. Mają one do 1 MB pamięci. Tagi Sony FeliCa są typowe dla tego typu. Więcej szczegółów na temat specyfikacji typów tagów można znaleźć w sekcji specyfikacji na stronie internetowej NFC Forum.

Ogólnie rzecz biorąc, wymiana NFC jest krótka. Osoba trzyma swoje urządzenie przy tagu lub innym urządzeniu, następuje krótka wymiana, a ona idzie dalej. Nie jest to protokół przeznaczony do długich wymian, ponieważ urządzenia muszą być dosłownie w kontakcie ze sobą. Podczas wysyłania dużych wiadomości użytkownik musi trzymać urządzenie w miejscu tak długo, jak długo trwa przesyłanie wiadomości. Może to być uciążliwe, więc ludzie zazwyczaj wolą używać NFC do wymiany funkcji między urządzeniami, a następnie przełączać się na WiFi lub Bluetooth w celu wymiany danych lub plików multimedialnych.

Oto typowy przykład, w którym NFC i WiFi mogą działać w tandemie: wyobraź sobie, że masz domowy odtwarzacz muzyki z obsługą NFC, który może synchronizować utwory ze smartfonem lub tabletem za pośrednictwem Wi-Fi z centralnego domowego serwera multimediów. Słuchasz albumu na domowym zestawie stereo, ale musisz wyjść do pracy. Dotykasz telefonu do zestawu stereo, a zestaw stereo informuje telefon za pośrednictwem NFC, który utwór jest odtwarzany i o której godzinie utworu. Następnie telefon sprawdza, czy ma ten utwór na swojej liście odtwarzania, a jeśli nie, pobiera go przez sieć komórkową lub Wi-Fi. Wychodzisz za drzwi, a gdy masz słuchawki na uszach, kontynuujesz odtwarzanie albumu w miejscu,

w którym skończyłeś.

Record Chunking

Jeśli konieczne jest wysłanie treści większej niż limit $^{232-1}$ bajtów, można podzielić ładunek na fragmenty i wysłać go w kilku rekordach. Gdy to zrobisz, ustawisz Chunk Flag (jeden z bitów flagi TNF) na 1 dla pierwszego podzielonego rekordu i wszystkich kolejnych rekordów, które są podzielone na fragmenty, z wyjątkiem ostatniego fragmentu. Nie można fragmentować zawartości w wielu komunikatach NDEF.

TNF jest ustawiany w pierwszym fragmencie rekordu. Kolejne fragmenty muszą używać TNF 06 (niezmieniony). Środkowe i końcowe fragmenty muszą mieć *długość typu* równą 0.

Długość ładunku każdego rekordu wskazuje ładunek *dla tego fragmentu.*

Posiadanie wiadomości przekraczającej 500 MB (czyli ^{232} bity) wydaje się mało prawdopodobne, więc chunking może nie być używany zbyt często. Jednak chunking może być również używany do dynamicznie generowanej zawartości, zwłaszcza gdy długość ładunku nie jest znana z góry.

Chunking jest używany stosunkowo rzadko. Biblioteki używane w tej książce nie implementują chunkingu. Parser Androida odczyta wiadomości podzielone na fragmenty i połączy je w logiczne rekordy NDEF.

Dodatkowe informacje

Więcej informacji na temat struktury NDEF, w tym przydatny zestaw testów do pisania własnych silników parsujących NDEF, można znaleźć na stronie specyfikacji NFC Forum.

NDEF w praktyce

Aby zobaczyć NDEF w praktyce, warto przyjrzeć się, jak robią to istniejące aplikacje. W tej sekcji będziesz musiał pobrać kilka istniejących aplikacji na swoje urządzenie, abyś mógł napisać z nimi kilka tagów i porównać ich pracę.

Jednym z najczęstszych zadań w wielu popularnych aplikacjach do pisania tagów jest meldowanie się w serwisie Foursquare. Po dotknięciu tagu sformatowanego za pomocą tego zadania urządzenie automatycznie połączy się z aplikacją społecznościową Foursquare i zamelduje użytkownika w danym miejscu. Jednak każda aplikacja zarządza tym zadaniem nieco inaczej, a różnice widoczne są w ich wynikach.

Do tego projektu potrzebne są następujące elementy:

- Telefon z systemem Android obsługujący technologię NFC
- Pięć tagów NFC
- Aplikacje wymienione poniżej (można je zainstalować na urządzeniu

bezpośrednio z Google Play na komputerze lub innym urządzeniu)

Oczekujemy również, że przeszedłeś przez rozdział 3 i zainstalowałeś całe oprogramowanie potrzebne do wykonania przykładów z tego rozdziału.

Aby uzyskać najlepsze wyniki na wielu urządzeniach, trzymaj się typów tagów NFC Forum; tagi Mifare Classic nie będą działać z wieloma nowszymi urządzeniami. Więcej informacji na temat tego, które tagi współpracują z którymi urządzeniami, znajduje się w sekcji "Dopasowanie typu urządzenia do tagu" na stronie 19. Unikaj tagów typu 2 (Mifare Ultralight), ponieważ nie mają one wystarczającej ilości pamięci dla tego projektu.

Do tego porównania wykorzystano następujące aplikacje:

- Wyzwalanie przez TagStand
- NFC TagWriter firmy NXP
- NFC Writer, również od TagStand
- TecTiles by Samsung (działa tylko w USA i Kanadzie)
- App Lancher NFC Tag Writer [sic] by vvakame

Będziesz także potrzebować NXP TagInfo do odczytywania tagów, Foursquare dla Androida i konta Foursquare.

Pamiętaj, aby zalogować się do aplikacji Foursquare przed przetestowaniem któregokolwiek z nich. Jeśli napotkasz jakiekolwiek problemy (takie jak miganie aplikacji Foursquare na ekranie i jej zniknięcie), wróć do aplikacji i zaloguj się ponownie. Jest to dobry przykład problemów, które mogą wystąpić całkowicie poza kontrolą użytkownika. Po przekazaniu ładunku do systemu Android, aplikacja (taka jak Foursquare) jest odpowiedzialna za to, co dzieje się dalej. Jeśli ta aplikacja nie poradzi sobie z tym z wdziękiem, możesz drapać się po głowie.

Nie martw się, jeśli w rzeczywistości nie znajdujesz się w lokalizacji, w której się meldujesz. Po pierwsze, każda aplikacja będzie musiała poprosić o pozwolenie na korzystanie z Foursquare przed faktycznym dokonaniem zameldowania, a zawsze możesz później usunąć zameldowanie na stronie internetowej Foursquare. Nie ma nic gorszego niż dezorientowanie znajomych co do miejsca pobytu podczas testowania NFC!

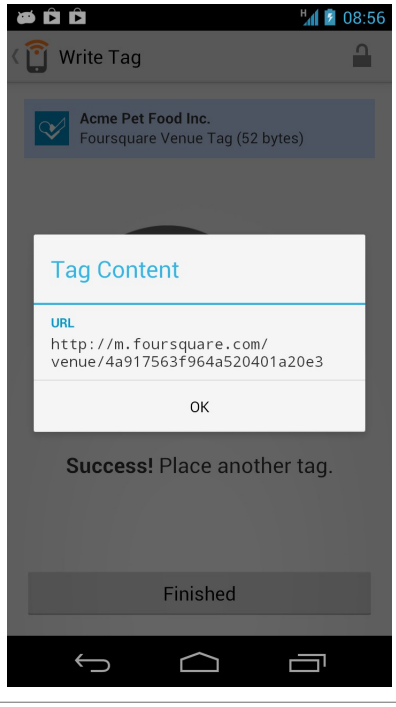
Proces pisania tagów dla każdej z tych aplikacji jest dość prosty, z wyjątkiem TagWriter. Oto podstawowe kroki:

Program uruchamiający zadania NFC
: Otwórz aplikację i kliknij przycisk + u góry, aby utworzyć nowe zadanie. Wybierz "NFC" z listy kategorii zadań. Nazwij zadanie, a następnie kliknij przycisk +. Z listy wybierz "Media społecznościowe", a następnie "Zamelduj się w miejscu na Foursquare". Wybierz miejsce, wpisując nazwę lub kliknij lupę, a aplikacja wyszuka pobliskie miejsca. Kliknij "OK", a następnie "Dodaj do zadania". Kliknij strzałkę w prawo, aby dodać tag. Umieść telefon nad tagiem, a aplikacja zapisze do niego dane.

TagStand Writer

Otwórz aplikację, kliknij "Foursquare Venue" i wybierz miejsce. Gdy ekran zmieni się na ekran pisania, kliknij nazwę miejsca, aby zobaczyć zawartość tagu. Zapisz

URL, jak pokazano na rysunku 4-3, ponieważ będzie on potrzebny dla aplikacji NXP TagWriter. Umieść telefon nad tagiem, a aplikacja zapisze do niego dane.



Rysunek 4-3. Gdy jesteś na ekranie pisania tagów w TagStand Writer, możesz kliknąć nazwę miejsca, aby zobaczyć pełny adres URL miejsca

NFC TagWriter

Otwórz aplikację i wybierz opcję "Utwórz, zapisz i przechowuj". Wybierz "Nowy" i z menu wybierz "URI". W polu opisu wpisz "Foursquare Check-in". Opis jest wymagany w tym ćwiczeniu, aby do tagu został zapisany rekord Smart Poster. Wprowadź miejsce Foursquare w następujący sposób: *http://m.foursquare.com/venue/ venueid*, gdzie *venueid* to długi ciąg szesnastkowy. Skopiuj go z tego, który wcześniej zapisałeś w TagStand Writer. Nie zapomnij wpisać http://. Kliknij "Dalej". Umieść telefon nad tagiem, a aplikacja zapisze do niego dane.

Samsung TecTiles

Upewnij się, że masz już zainstalowaną aplikację Foursquare. Otwórz aplikację TecTiles i kliknij "Nowe zadanie". Nadaj mu nazwę, a następnie kliknij "OK". Kliknij "Dodaj", aby utworzyć nowe zadanie. Kliknij "Aplikacja". Wybierz Foursquare z listy aplikacji. Kliknij "Zapisz do tagu". Umieść telefon nad tagiem, a aplikacja zapisze do niego dane.

AppLauncher NFC
: Otwórz aplikację i wybierz Foursquare. Umieść telefon nad tagiem, a aplikacja zapisze do niego dane.

Po napisaniu i oznaczeniu pięciu tagów wypróbuj je. Zamknij wszystkie aplikacje i umieść telefon nad znacznikiem. Powinny one reagować w sposób opisany w Tabeli 4-1.

Następnie otwórz aplikację NXP TagInfo. Umieść telefon nad tagiem, a aplikacja odczyta tag. Wybierz NDEF i możesz zobaczyć szczegóły NDEF dowolnego tagu (jak pokazano w Tabeli 4-1).

Tabela 4-1. Dane wyjściowe każdej aplikacji

Aplikacja	Rekord	TNF	Typ rekordu	Ładunek	Działanie
Zadanie NFC Launcher	1	MIME	x/nfctl	enZ:Foursquare;c:4a917563f964a520401a20e3	Próby uruchomienie Zadanie Aplikacja uruchamiająca; jeśli udany, przekazuje adres URL do Aplikacja Foursquare
	2	Zewnętrzne	android.com:pkg	com.jwsoft.nfcactionlauncher	
Tagstand Pisarz:	1	Cóż-Znany	U	*http://m.foursquare.com/venue/4a917563f964a520401a20e3*	Uruchomienia Foursquare aplikacja, przenosi użytkownika do sprawdzenia miejsca-na ekranie
NXP TagWriter	1	Cóż-Znany	Sp		Uruchomienia Foursquare aplikacja, przenosi użytkownika do sprawdzenia miejsca-na ekranie
	1.1		U	*http://m.foursquare.com/venue/4a917563f964a520401a20e3*	
	1.2		T	Zameldowanie w serwisie Foursquare	
Samsung TecTiles	1	Cóż-Znany	U	tectile://www/samsung.com/tectiles	Próby uruchomienie

					TecTiles aplikacja
	2	Cóż-Znany	T	-enTask···Foursquare-com.joelapenna...[a]	
Aplikacja Launcher NFC	1	Zewnętrzne	android.com:pkg	com.joelapenna.foursquared	Uruchomienia Aplikacja Foursquare tylko

[a] Komunikat Samsung TecTiles zawiera niedrukowalne znaki. Rzeczywiste znaki nie są ważne i zostały zastąpione środkową kropką. Linia została również obcięta wielokropkiem. Możesz zobaczyć pełny strumień bajtów dla tych tagów, zapisując tag z TecTiles i odczytując go za pomocą NXP TagInfo.

Jak widać, każda aplikacja wykonuje swoją pracę w inny sposób. Istnieją jednak cztery podstawowe podejścia przedstawione tutaj:

- Uruchom aplikację Foursquare i pozwól użytkownikowi zrobić resztę (App Launcher NFC)
- Wyślij URI i pozwól systemowi operacyjnemu zrobić resztę (Tagstand Writer)
- Uruchom oryginalną aplikację, która z kolei uruchomi aplikację Foursquare (NFC Task Launcher, Samsung TecTiles).
- Wyślij inteligentny plakat (NXP TagWriter)

Pierwsza metoda używa tylko jednego rekordu NDEF, z TNF ustawionym na "External". Typ rekordu to wtedy Android Application Record dla aplikacji, którą chcesz uruchomić, a zawartość to rzeczywista nazwa aplikacji, np:

```
TNF: Zewnętrzny
Typ rekordu: android.com:pkg
com.joelapenna.foursquared
```

Korzystając z pierwszej metody, mówisz Androidowi, która aplikacja ma zostać uruchomiona.

Druga metoda również wykorzystuje tylko jeden rekord NDEF, z TNF ustawionym na "Well-Known" i typem rekordu ustawionym na "U" dla URI. Ponownie, treścią jest rzeczywisty adres, np:

```
TNF: Znany typ
rekordu: U
http://m.foursquare.com/venue/4a917563f964a520401a20e3
```

Korzystając z drugiej metody, mówisz Androidowi URI rzeczy, którą chcesz otworzyć i pozwalasz systemowi operacyjnemu zdecydować, która aplikacja najlepiej ją otworzy. To trochę tak, jakby pozwolić systemowi Windows zdecydować, która aplikacja otworzy plik z określonym rozszerzeniem. Gdyby Foursquare nie było na twoim urządzeniu, Google Play otworzyłby się, aby obsłużyć te adresy URL.

Trzecia metoda wykorzystuje komunikat NDEF składający się z dwóch rekordów NDEF. Zarówno w przypadku NFC Task Launcher, jak i Samsung TecTiles, oryginalna aplikacja obsługuje odczyt tagu, a następnie uruchamia Foursquare. NFC Task Launcher używa rekordu typu MIME, który zawiera informacje o miejscu Foursquare i zewnętrzny rekord AAR, który zapewnia, że aplikacja jest zainstalowana. TecTiles przyjmuje podobne podejście z inną implementacją. TecTiles używa rekordu URI z niestandardowym adresem URL *tectile://* do uruchomienia aplikacji. Koduje informacje Foursquare w drugim rekordzie tekstowym. Niestety, TecTiles uruchamia tylko aplikację; nie przechowuje informacji o miejscu. Obie aplikacje używają filtrów intencji, aby uruchomić się po zeskanowaniu ich tagu. NFC Task Launcher rejestruje się dla typu MIME `x/nfctl`. TecTiles rejestruje się dla swojego niestandardowego *tectile://* URI. Więcej informacji na temat filtrów intencji znajduje się w sekcji "Android's

Tag Dispatch System" na stronie 89.

Czwarta metoda wykorzystuje rekord Smart Poster. Smart Posters to bardziej złożony typ rekordu NDEF, w którym ładunek jest w rzeczywistości innym komunikatem NDEF. Komunikat

osadzony w ładunku Smart Poster zawiera dwa własne rekordy NDEF, URI i rekord tekstowy. Ponieważ rekordy Smart Poster mają wiele rekordów, mogą dostarczać dodatkowych informacji o URI, takich jak tytuł, ikona lub sugerowane akcje przetwarzania.

Można zauważyć, że niektóre aplikacje, takie jak TecTiles i NFC Task Launcher, zapisują rekordy aplikacji Androida, aby uruchomić własną aplikację, a nie Foursquare. Następnie ich aplikacja uruchamia Foursquare. Przypuszczalnie pozwala im to śledzić, kiedy ich aplikacja jest używana, nawet jeśli końcowym rezultatem jest otwarcie innej aplikacji. Jest to bardziej skomplikowane, ale prawdopodobnie pozwala na gromadzenie danych na temat korzystania z aplikacji.

Aplikacja do pisania tagów: Foursquare Check-In

W tej sekcji napiszesz własną aplikację do zapisu tagów, aby lepiej zrozumieć, jak to działa w praktyce. Ta aplikacja jest bardzo prosta; szuka tagu, który można sformatować jako tag NDEF, a jeśli go znajdzie, zapisuje wiadomość NDEF do tagu.

Do tego projektu potrzebne są następujące elementy:

- Telefon z systemem Android obsługujący technologię NFC
- Pięć tagów RFID
- Foursquare dla systemu Android i konto Foursquare

Zacznij od utworzenia nowego projektu, tak jak w rozdziale 3. Użyj Cordova, aby utworzyć projekt, dodać platformę Android i zainstalować wtyczkę:

```
$ cordova create ~/FoursquareCheckin com.example.checkin FoursquareCheckin ❶
$ cd ~/FoursquareCheckin ❷
$ cordova platform add android
$ cordova plugin add https://github.com/chariotsolutions/phonegap-nfc
```

❶ Użytkownicy systemu Windows powinni wpisać `%userprofile%\FoursquareCheckin` zamiast `~/FoursquareCheckin`.

❷ Użytkownicy systemu Windows powinni wpisać `/d %userprofile%\FoursquareCheckin` zamiast `~/FoursquareCheckin`.

Teraz możesz napisać swoją aplikację, edytując pliki HTML i JavaScript. Plik *index.*html znajduje się w katalogu *www* katalogu aplikacji, który właśnie utworzyłeś, a *index.js* w *www/js*. Otwórz je oba i usuń wszystko, aby rozpocząć własną aplikację od zera. Zacznij od pliku *index.html w następujący sposób*:

```
<!DOCTYPE html>

<html>
```

```
<head>
   <title>Foursquare Check-In Tag Writer</title>
```

```
    <style>body { margin: 20px }</style>.
  </head>
  <body>
    <p>Foursquare Check-In Tag Writer</p>

    <div class="app">.
      <div id="messageDiv">Nie znaleziono tagu</div>
    </div>

    <script type="text/javascript" src="cordova.js"></script>.
    <script type="text/javascript" src="js/index.js"></script>.
    <script type="text/javascript">
          app.initialize();
      </script>
  </body>
</html>
```

Zapisywanie rekordu NDEF do znacznika

Następnie napiszesz plik *index.js*, aby sformatować rekord NDEF i napisać wiadomość NDEF do każdego napotkanego tagu. Na razie zachowasz prostotę i zakodujesz na sztywno większość parametrów. Widziałeś wiele metod wyzwalania zameldowania w Foursquare; na początek zrób najprostszą rzecz: uruchom aplikację Foursquare za pomocą Android Application Record.

Zacznij od zmiennej, która będzie zapisywana, gdy pojawi się tag:

```
var app = {
  messageToWrite: [],      // wiadomość do zapisania w następnym zdarzeniu NFC
```

Następnie pojawia się funkcja `initialize()`, aby rozpocząć pracę i funkcja `bindEvents()`, aby skonfigurować detektor zdarzeń do wykrywania, kiedy urządzenie jest gotowe:

```
  // Konstruktor aplikacji
  initialize: function() {
    this.bindEvents();
console.log("Uruchamianie aplikacji Foursquare Checkin");
  },
  /*
    powiązać wszelkie zdarzenia, które są wymagane podczas uruchamiania, ze
    słuchaczami:
  */
  bindEvents: function() {
    document.addEventListener('deviceready', this.onDeviceReady, false);
  },
```

Następnie pojawia się handler do wyczyszczenia ekranu i dodania event listenera do nasłuchiwania wykrytych tagów:

```
  /*
    uruchamia się, gdy urządzenie jest gotowe do interakcji z użytkownikiem:
  */
```

```
        onDeviceReady: function() {
```

```
        app.clear();

        nfc.addTagDiscoveredListener(
            app.onNfc,              // tag został pomyślnie zeskanowany
            function (status) {     // słuchacz został pomyślnie zainicjowany
                app.makeMessage();
                app.display("Dotknij tagu NFC, aby zapisać dane");
            },
            function (error) {      // słuchacz nie został zainicjowany
                app.display("Nie udało się zainicjować czytnika NFC "
                    + JSON.stringify(error));
            }
        )
    },
```

Obsługa zdarzenia NFC, `onNfc()`, zapisze dane do tagu w następujący sposób:

```
    /*
        wywoływana po odczytaniu tagu NFC:
    */
    onNfc: function(nfcEvent) {
        app.writeTag(app.messageToWrite);
    },
```

Następnie pojawiają się funkcje `display()` i `clear()`, tak jak je napisałeś w "PhoneGap Meets NFC: NFC Reader" na stronie 40:

```
    /*
        dołącza @message do elementu div wiadomości:
    */
    display: function(message) {
        var label = document.createTextNode(message),
            lineBreak = document.createElement("br");
        messageDiv.appendChild(lineBreak);      // dodanie podziału wiersza
        messageDiv.appendChild(label);          // dodanie tekstu
    },
    /*
        czyści div komunikatu:
    */
    clear: function() {
         messageDiv.innerHTML = "";
    },
```

Poniższe metody funkcji `makeMessage()` i `writeTag()` wykorzystują funkcje z dwóch obiektów zdefiniowanych przez wtyczkę NFC. Są to obiekt NFC, który zapewnia dostęp do czytnika NFC urządzenia, oraz obiekt NDEF, który definiuje i formatuje rekordy i wiadomości NDEF.

```
    makeMessage: function() {
        // Złożenie elementów komunikatu NDEF:
        var tnf = ndef.TNF_EXTERNAL_TYPE,           // NDEF Type Name
            Format recordType = "android.com:pkg",  // NDEF Record Type
            payload = "com.joelapenna.foursquared", // zawartość rekordu
```

```
            rekord,                    // obiekt rekordu NDEF
            message = [];              // Komunikat NDEF do przekazania do writeTag()

        // utworzyć rzeczywisty rekord NDEF:
        record = ndef.record(tnf, recordType, [], payload);
        // umieszczenie rekordu w tablicy
        komunikatów: message.push(record);
        app.messageToWrite = message;
    },

    writeTag: function(message) {
        // zapisanie rekordu do tagu:
        nfc.write(
            message,           // zapisuje sam rekord do tagu
            function () {      // po zakończeniu uruchom tę funkcję zwrotną:
                app.display("Zapisano dane do tagu.");     // zapis do elementu div
                wiadomości
            },
            // ta funkcja jest uruchamiana, jeśli polecenie zapisu nie
            powiedzie się:
            function (reason) {
                alert("Wystąpił problem " + powód);
            }
        );
    }
}; // koniec aplikacji
```

Wiesz już, że rekord NDEF składa się z TNF, typu rekordu i ładunku. Obiekt NDEF posiada funkcje do tworzenia rekordów NDEF#Records. Komunikat NDEF#Message jest po prostu tablicą rekordów NDEF#Records. Aby utworzyć nowy rekord NDEF, należy przekazać cztery parametry:

Format nazwy typu
: Wartość 3-bitowa. Wartości TNF są zawarte jako stałe w obiekcie NDEF, więc można się do nich odwoływać za pomocą tych stałych.

Typ rekordu
: Łańcuch lub tablica bajtów zawierająca od zera do 255 bajtów, reprezentująca typ rekordu. Obiekt NDEF ma wbudowane stałe dla wielu z nich, choć nie wszystkie, jak widać w tym przykładzie.

Identyfikator rekordu
: Ciąg znaków lub tablica bajtów zawierająca od zera do 255 bajtów. Jak przeczytałeś wcześniej, identyfikatory rekordów są opcjonalne, więc możesz użyć pustej tablicy, ale ta wartość nie może być zerowa.

Ładunek
: Ciąg lub tablica zawierająca od zera do (2^{32-1}) bajtów. Ponownie, można użyć pustej tablicy, ale nie może ona mieć wartości null.

W funkcji `makeMessage()`, TNF to `TNF_EXTERNAL_TYPE`, przy użyciu stałych

zdefiniowanych w obiekcie NDEF; typem rekordu jest `android.com:pkg`, co oznacza, że ładunek będzie rekordem aplikacji Android; identyfikator rekordu nie jest określony, więc rekord

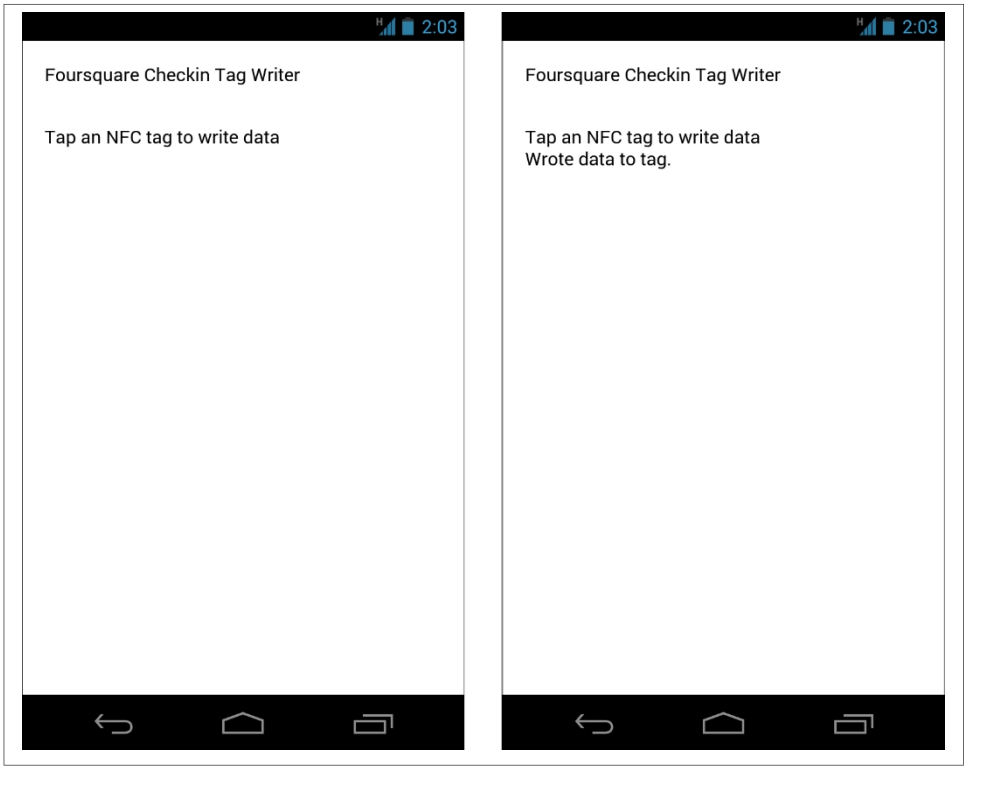
wysyłana jest pusta tablica; a ładunkiem jest nazwa aplikacji do uruchomienia, `com.joe lapenna.foursquared`.

Zapisz pliki *index.html* i *index.js, a* następnie zmień katalogi na katalog główny nowej aplikacji i uruchom ją:

```
cordova run
```

Po uruchomieniu aplikacji i dotknięciu urządzenia do tagu, powinien pojawić się identyfikator tagu, jak pokazano po lewej stronie rysunku 4-4. Kliknij łącze, a aplikacja zapisze się do tagu i wyświetli powiadomienie widoczne po prawej stronie rysunku 4-4. Zamknij aplikację, stuknij urządzeniem w tag, co spowoduje uruchomienie aplikacji Foursquare.

Pełny kod źródłowy można znaleźć na GitHub.



Rysunek 4-4. Aplikacja Foursquare do meldowania się; oczekiwanie na tag (po lewej) i zapisywanie do tagu (po prawej)

Zapisywanie różnych typów rekordów

Twoja aplikacja może teraz napisać tag, aby uruchomić inną aplikację z tego tagu, ale nie robi wszystkich rzeczy, które widziałeś w poprzednich przykładach. Byłoby wspaniale, gdybyś mógł emulować wszystkie pięć innych aplikacji, dla porównania. Aby to zrobić, utwórz nowy projekt o nazwie "Four- squareAdvanced". Zainstaluj wtyczkę NFC, a następnie skopiuj pliki *index.html* i *index.js* z poprzedniego przykładu aplikacji Foursquare Check-In:

```
$ cordova create ~/FoursquareAdvanced com.example.advanced FoursquareAdvanced ❶
$ cd ~/FoursquareAdvanced ❷
$ cordova platform add android
$ cordova plugin add https://github.com/chariotsolutions/phonegap-nfc
$ cp ~/FoursquareCheckin/www/index.html ~/FoursquareAdvanced/www/.
$ cp ~/FoursquareCheckin/www/js/index.js ~/FoursquareAdvanced/www/js/.
```

❶ Użytkownicy systemu Windows powinni wpisać `%userprofile%\FoursquareAdvanced` zamiast `~/FoursquareAdvanced`.

❷ Użytkownicy systemu Windows powinni wpisać `/d %userprofile%\FoursquareAdvanced` zamiast `~/FoursquareAdvanced`.

Zacznij od zmodyfikowania strony *index.*html, dodając formularz z menu opcji, które pozwala użytkownikowi wybrać aplikację, którą chce emulować. Oto nowa strona *index.html:*

```
<!DOCTYPE html>

<html>
   <head>
    <title>Foursquare Check-In Tag Writer - Advanced</title>
    <style>
        body { margin: 20px }
    </style>
   </head>
   <body>
    <p>Foursquare Check-In Tag Writer - Advanced</p>
    <div class="app">.
      <form>
       Napisz tag taki jak: <br />
       <select id="appPicker">
         <option value="1">NFC Task Launcher</option>.
         <option value="2">TagStand Writer</option>.
         <option value="3">NXP TagWriter</option>.
         <option value="4">Samsung TecTiles</option>
         <option value="5">App Launcher NFC</option>.
       </select>
      </form>
      <div id="messageDiv"></div>
    </div>
    <script type="text/javascript" src="cordova.js"></script>.
```

```
<script type="text/javascript" src="js/index.js"></script>.
<script type="text/javascript">.
```

```
        app.initialize();
      </script>
    </body>
</html>
```

Gdy użytkownik wybierze jedną z aplikacji w menu, otrzymasz wygodny element formularza, który możesz odczytać, aby określić sposób formatowania tagu. Zapisz plik *index.html* i otwórz plik *index.js*. *Będziesz musiał* zmodyfikować `makeMessage()`, aby to obsłużyć. Nowa funkcja będzie emulować aplikacje, które widziałeś tutaj, konstruując kilka różnych typów komunikatów NDEF. Najpierw dodaj nową zmienną lokalną, aby odczytać element formularza HTML i dowiedzieć się, którą aplikację użytkownik chce emulować. Pozostałe zmienne lokalne mają te same nazwy, ale tym razem ich wartości będą generowane w locie:

```
makeMessage: function() {
  // pobiera typ aplikacji, który użytkownik chce emulować z formularza HTML:
  var appType = parseInt(appPicker.value, 10),
      tnf,              // Format nazwy typu NDEF
      recordType,       // ładunek  typu rekordu
      NDEF,             // zawartość rekordu
      rekord,           // obiekt rekordu NDEF
      message = [];     // Komunikat NDEF do przekazania do writeTag()
```

Reszta funkcji `makeMessage()` zostanie całkowicie zmieniona. Po zmiennych lokalnych należy sformatować inną wiadomość NDEF w zależności od tego, którą aplikację się emanuje. Ponieważ formularz HTML zwraca liczbę całkowitą z menu opcji, można użyć tego wyniku w instrukcji switch-case w następujący sposób. Widać, że każdy przypadek zapisuje te same rekordy, co jedna z poprzednich aplikacji. Przypadek 1 tworzy rekord MIME media zawierający instrukcje dla aplikacji Task Launcher, a następnie rekord aplikacji Android, aby powiedzieć Androidowi, którą aplikację uruchomić:

```
  switch (appType) {
     case 1: // jak NFC Task Launcher
        // formatowanie rekordu nośnika MIME:
        recordType = "x/nfctl";
        payload = "enZ:Foursquare;c:4a917563f964a520401a20e3";
        record = ndef.mimeMediaRecord(recordType, payload);
        message.push(record); // umieszczenie rekordu w wiadomości

        // formatowanie rekordu aplikacji Android:
        tnf = ndef.TNF_EXTERNAL_TYPE;
        recordType = "android.com:pkg";
        payload = "com.jwsoft.nfcactionlauncher";
        record = ndef.record(tnf, recordType, [], payload);
        message.push(record); // umieść rekord w komunikacie break;
```

Przypadek 2 tworzy rekord typu Well-Known z RTD ustawionym na URI. Ponieważ identyfikatory URI mają standardowy format, specyfikacja NDEF zawiera kody identyfikatorów URI, które mogą być używane zamiast niektórych standardowych nagłówków URI w celu skrócenia ładunku. Na przykład tabela 4-2 przedstawia kilka

pierwszych kodów identyfikatorów URI.

Tabela 4-2. Niektóre kody identyfikatorów URI

Kod identyfikatora URI (UIC)	Co oznacza
0x00	Nothing, używane dla niestandardowych nagłówków
0x01	*http://www.*
0x02	*https://www.*
0x03	*http://*

Inne nagłówki URI, do których jesteś przyzwyczajony, takie jak *ftp://* i *mailto:* oraz *file:///*, są dołączone z własnymi numerami. Pełna lista znajduje się w dokumencie URI Record Type Definition, będącym częścią specyfikacji NFC na stronie NFC Forum. Jest ona również wymieniona w Załączniku A.

Aby dodać UIC, należy przekonwertować ciąg URI na tablicę bajtów i umieścić UIC na jej początku. W przypadku 2 należy dodać 0x03, UIC dla http://:

```
case 2: // jak Tagstand Writer
   // formatuje rekord URI jako typ Well-Known:
   tnf = ndef.TNF_WELL_KNOWN;
   recordType = ndef.RTD_URI; // dodanie typu rekordu URI
   // przekonwertować na tablicę bajtów:
   payload = nfc.stringToBytes(
      "m.foursquare.com/venue/4a917563f964a520401a20e3");
   // dodać kod identyfikatora URI dla "http://":
   payload.unshift(0x03);
   record = ndef.record(tnf, recordType, [], payload);
   message.push(record); // umieszczenie rekordu w komunikacie
   break;
```

Przypadek 3 jest wyjątkowy, ponieważ tworzy wiadomość Smart Poster. Smart Posters są wyjątkowe wśród znanych typów Well-Known, ponieważ są to rekordy zawierające wiadomości NDEF. Jak widać w następnym przykładzie, ładunek rekordu Smart Poster jest konstruowany jako tablica rekordów. Ta tablica to wiadomość.

W rozdziale 5 zobaczysz, jak wyodrębnić rekord Smart Poster, traktując jego zawartość jako wiadomość i wyodrębniając tę wiadomość rekursywnie:

```
case 3: // jak NXP TagWriter
   // Ładunkiem rekordu Smart Poster jest komunikat NDEF
   // więc utwórz tablicę dwóch rekordów w następujący sposób:
   var smartPosterPayload = [
      ndef.uriRecord(
         "http://m.foursquare.com/venue/4a917563f964a520401a20e3"),
      ndef.textRecord("foursquare checkin"),
   ];

   // Utworzenie rekordu Smart Poster z tablicy:
   record = ndef.smartPoster(smartPosterPayload);
   // umieszczenie wpisu Smart Poster w wiadomości:
   message.push(record);
   przerwa;
```

Przypadek 4 konstruuje rekord URI i rekord tekstowy zawierający pewne dane binarne. Jak widać w przypadku 2, należy przekonwertować identyfikator URI na tablicę bajtów i umieścić kod identyfikatora URI z przodu tablicy. W tym przypadku, ponieważ Samsung używa niestandardowego nagłówka URI (*tectile://*), należy użyć kodu identyfikatora URI 0x00, który zapisuje URI dokładnie tak, jak napisano. TecTiles zapisuje również drugi rekord z TNF 01 (Well-Known type) i typem "T".

Samsung używa również zastrzeżonego tokena w środku ładunku. Można to wykryć, odczytując tag napisany w aplikacji TecTiles za pomocą aplikacji NXP TagReader. Jest on tutaj zduplikowany, aby wiadomość była sformatowana tak, jak chce tego aplikacja Samsung:

```
przypadek 4: // jak TecTiles
    // sformatuj rekord jako typ Well-Known
    tnf = ndef.TNF_WELL_KNOWN;
    recordType = ndef.RTD_URI; // dodanie typu rekordu URI
    var uri = "tectiles://www.samsung.com/tectiles";
    payload = nfc.stringToBytes(uri);
    var id = nfc.stringToBytes("0");
    // Identyfikator URI 0x00, ponieważ nie ma identyfikatora dla
    "tectile://":
    payload.unshift(0x00);
    record = ndef.record(tnf, recordType, id, payload);
    message.push(record); // umieszczenie rekordu w wiadomości

    // rekord tekstowy z danymi
    binarnymi tnf =
    ndef.TNF_WELL_KNOWN;
    recordType = ndef.RTD_TEXT;
    payload = [];
    // długość kodu języka
    payload.push(2);
    // kod języka
    payload.push.apply(payload, nfc.stringToBytes("en"));
    // Nazwa zadania
    payload.push.apply(payload, nfc.stringToBytes("Task"));
    // 4-bajtowy token zastrzeżony dla TecTiles:
    payload.push.apply(payload, [10, 31, 29, 19]);
    // Nazwa aplikacji
    payload.push.apply(payload, nfc.stringToBytes("Foursquare"));
    // terminator NULL
    payload.push(0);
    // Aktywność do uruchomienia
    payload.push.apply(payload,
       nfc.stringToBytes("com.joelapenna.foursquared.MainActivity"));
    // terminator NULL
    payload.push(0);
    // Application packageName
    payload.push.apply(payload,
       nfc.stringToBytes("com.joelapenna.foursquared"));
    id = nfc.stringToBytes("1");
    record = ndef.record(tnf, recordType, id, payload);
```

```
            message.push(record); // umieszczenie rekordu w wiadomości
            przerwa;
```

Rekord TecTiles jest zbyt duży dla niektórych tagów, takich jak Mifare Ultralight, który ma ograniczoną pojemność.

Przypadek 5 tworzy rekord aplikacji na Androida, aby bezpośrednio otworzyć Foursquare. Jest taki sam jak wszystkie inne rekordy aplikacji na Androida, które już utworzyłeś. Jest to zewnętrzny TNF, z typem rekordu `android.com:pkg` i nazwą aplikacji jako ładunkiem:

```
        case 5: // jak App Launcher NFC
            // formatowanie rekordu aplikacji Android:
            tnf = ndef.TNF_EXTERNAL_TYPE;
            recordType = "android.com:pkg";
            payload = "com.joelapenna.foursquared";
            record = ndef.record(tnf, recordType, [], payload);
            message.push(record); // umieść rekord w komunikacie break;
    } // koniec instrukcji switch-case
```

Na koniec zakończ funkcję `makeMessage()`, ustawiając komunikat na write i powiadamiając użytkownika:

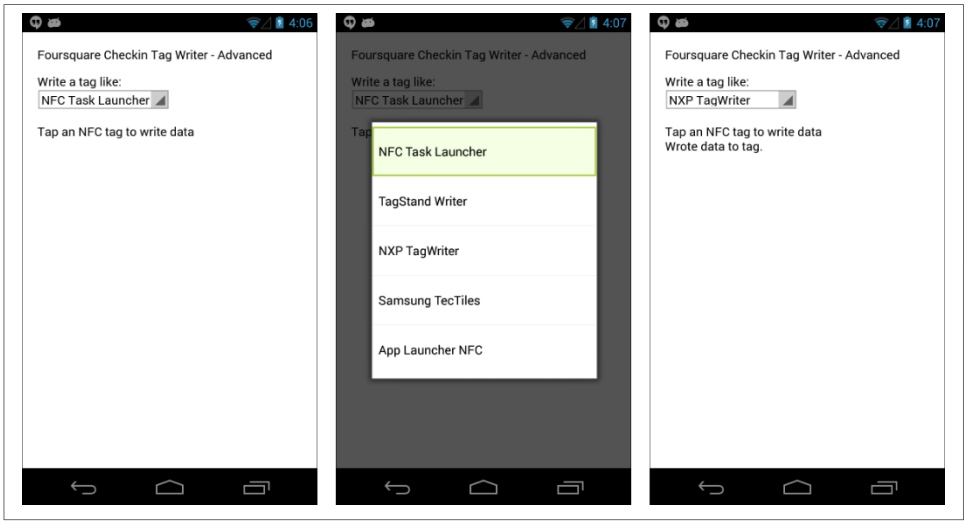
```
    app.messageToWrite = message;
    app.display("Dotknij tagu NFC, aby zapisać
    dane");
}, // koniec makeMessage()
```

Po wprowadzeniu tych zmian możesz zapisać plik *index.js* i uruchomić aplikację (upewnij się, że nadal masz w aplikacji funkcję `writeTag()` z poprzedniego przykładu):

```
$ cordova run
```

Gdy to zrobisz, pojawi się rozwijane menu, które pozwala emulować dowolną z aplikacji widocznych w tabeli 4-1. Aby z niego skorzystać, wybierz aplikację do emulacji z menu, a następnie stuknij telefon w znacznik. Rysunek 4-5 przedstawia aplikację.

Pełny kod źródłowy można znaleźć na GitHub.



Rysunek 4-5. Aplikacja Foursquare Check-In Advanced; ekran początkowy (po lewej), pokazujący menu opcji (w środku) i po zapisaniu do tagu (po prawej)

Podsumowanie funkcji pomocniczych PhoneGap-NFC NDEF

Jak widać, ta aplikacja obejmuje teraz wiele funkcji biblioteki PhoneGap-NFC, emulując różne podejścia do tego samego zadania. W zawartym w niej obiekcie NDEF znajduje się kilka funkcji pomocniczych, których można użyć do zapisywania rekordów NDEF. Główną z nich jest `ndef.record()`, która wymaga podania TNF, typu rekordu, ID i ładunku:

```
// Wywołujący określa TNF i typ rekordu
record = ndef.record(tnf, recordType, id, payload);
// przykład:
record = ndef.record(ndef.TNF_EXTERNAL_TYPE, "android.com:pkg", [],
   "com.joelapenna.foursquared");
```

Istnieje również pomocnik dla rekordów multimedialnych MIME, który pobiera tylko typ MIME i ładunek:

```
// TNF: MIME media (02)
record = ndef.mimeMediaRecord(mimeType, payload);
// przykład:
record = ndef.mimeMediaRecord("text/json", '{"answer": 42}');
```

Pomocnik URI tworzy rekord przy użyciu TNF 01, Well-Known i typu rekordu "U" w celu wysłania URI, jak pokazano w Przypadku 3 "Pisanie różnych typów rekordów" na stronie 66. Pomocnik skróci identyfikator URI za pomocą kodu identyfikatora URI. W tym przykładzie *http://* zostanie zapisany w tagu jako 0x03, po którym nastąpi *m.foursquare.com*:

```
// TNF: Znany typ(01), RTD: URI ("U")
record = ndef.uriRecord(uri);
```

```
// przykład:
record = ndef.uriRecord("http://m.foursquare.com/");
```

Pomocnik rekordu tekstowego tworzy rekordy tekstowe przy użyciu Well-Known TNF i definicji typu rekordu tekstowego. Jeśli nie określono języka, domyślnie jest to angielski:

```
// TNF: Znany typ(01), RTD: Tekst ("T")
record = ndef.textRecord(text, language);
// przykład:
// domyślnie angielski, ponieważ nie określono języka:
record = ndef.textRecord("Jak się masz?");
```

Pomocnik Smart Poster konstruuje Smart Poster z innych rekordów:

```
//TNF: Znany typ (01), RTD: Inteligentny plakat ("Sp")
record = ndef.smartPoster(ndefMessage);
// przykład:
record = ndef.smartPoster (
   // Rekord URI:
   ndef.uriRecord("http://m.foursquare.com/venue/4a917563f964a520401a20e3"),
   // zapis tekstowy:
   ndef.textRecord("foursquare checkin")
);
```

Istnieje również pomocnik, którego nie widziałeś w tym rozdziale, pomocnik pustego rekordu. Tworzy on pusty rekord do wypełnienia:

```
// TNF: Pusty (00)
record = ndef.emptyRecord();
// przykład:
record = ndef.emptyRecord(); // jest pusty!
```

Za pomocą tych funkcji można skonstruować wszystkie typy komunikatów NDEF, które pojawią się w dalszej części tej książki.

Wnioski

Zagłębimy się bardziej w rekordy i komunikaty NDEF w kolejnych rozdziałach. Kluczowe pojęcia, które należy zabrać z tego wprowadzenia, są jednak następujące:

Tagi w formacie NFC zawierają *komunikaty* NDEF. Komunikaty NDEF składają się z jednego lub więcej *rekordów* NDEF. Będziesz kuszony, aby mówić o tagach, wiadomościach i rekordach zamiennie, ale nie rób tego. W dalszej części książki będziesz przekazywać wiadomości z urządzenia do urządzenia bez tagów.

Wszystkie rekordy NDEF mają swój typ. W specyfikacjach jest on czasami nazywany typem ładunku, czasami typem rekordu, a innym razem po prostu typem. Format nazwy typu (TNF) kategoryzuje typy ogólnie na kilka obszarów i mówi, jak interpretować typ:

Dla TNF 01 (dobrze znany)
: Specyfikacja NFC Forum Record Type Definition (RTD) określa, jakich typów należy używać.

Dla TNF 02 (MIME)
: MIME RFC (RFC 2046) określa, jakie typy są prawidłowe.

Dla TNF 03 (URI)
: URI RFC (RFC 3986) mówi, jak tworzyć prawidłowe typy.

Dla TNF 04 (zewnętrzny)
: Specyfikacja NFC Forum RTD zawiera informacje na temat tworzenia własnych typów.

Istnieje osiem formalnych formatów nazw typów komunikatów NDEF, ale większość pracy wykonywana jest tylko z trzema: Well-Known, MIME media i External. Well-Known TNF zawiera szereg przydatnych definicji typów rekordów, w tym wiadomości tekstowych, URI, Smart Posters i różnych nośników potrzebnych do przekazywania peer-to-peer. Więcej informacji na temat komunikatów peer-to-peer znajduje się w rozdziale 8. Wiadomości tekstowe i URI będą używane w całej tej książce. MIME media TNF obejmuje wszystkie typy mediów internetowych i można go używać do tworzenia niestandardowych typów, jak zobaczysz w rozdziale 6. Zewnętrzny TNF jest używany dla rekordów takich jak Android Application Record.

W tej książce nie zobaczysz wielu typów Smart Poster. Jak widać w przypadku Smart Poster w sekcji "Pisanie różnych typów rekordów" na stronie 66, ładunek rekordu Smart Poster jest sam w sobie komunikatem NDEF. Tak więc, aby odczytać Smart Poster, musisz najpierw przeanalizować wiadomość, która go zawiera, a następnie musisz przeanalizować wiadomość, którą on zawiera. Uważamy, że jest to zbędne dla większości aplikacji i że lepszą praktyką jest po prostu użycie wielu rekordów w komunikacie NDEF.

Istnieje wiele sposobów na osiągnięcie tych samych celów za pomocą komunikatów NDEF. Sposób odbioru wiadomości zależy od systemu operacyjnego urządzenia, które ją odbiera. W następnym rozdziale przyjrzymy się, jak Android oferuje kilka różnych sposobów filtrowania typów NDEF do aplikacji.

ROZDZIAŁ 5

Odsłuchiwanie komunikatów NDEF

Każda dobrze zaprojektowana aplikacja opiera się na dobrym nasłuchiwaniu odpowiednich zdarzeń, niezależnie od tego, czy są to zdarzenia wejściowe użytkownika, zdarzenia sieciowe czy inne. Nie inaczej jest w przypadku aplikacji NFC. Aplikacje NFC zawsze nasłuchują komunikatów NDEF, analizują i filtrują typ i treść wiadomości, a następnie podejmują odpowiednie działania. W rozdziale 4 przedstawiono sposób pisania wiadomości NDEF za pomocą wtyczki PhoneGap-NFC, ale nie zagłębialiśmy się w część nasłuchiwania.

Istnieją dwa główne sposoby nasłuchiwania wiadomości NDEF: możesz zaprogramować swoją aplikację, aby nasłuchiwała ich, gdy jest to aplikacja na pierwszym planie, lub możesz pozwolić systemowi operacyjnemu urządzenia na nasłuchiwanie i wywoływanie aplikacji, gdy zobaczy wiadomość, która jest istotna dla Twojej aplikacji. Wtyczka PhoneGap-NFC oferuje kilka zdarzeń NFC, na które można zaprogramować aplikację, aby nasłuchiwała, a Android oferuje *system wysyłania tagów*, który pozwala powiedzieć systemowi operacyjnemu, które tagi są najbardziej interesujące. Aplikacje NFC mogą korzystać z systemu wysyłania tagów, pisząc *filtry intencji* w manifeście Androida, które informują system operacyjny, które tagi mają być kierowane do aplikacji. Mogą również wyraźnie nasłuchiwać niektórych lub wszystkich tagów, zastępując filtry intencji innych aplikacji korzystających z *systemu wysyłania na pierwszym planie*. W tym rozdziale dowiesz się nieco więcej o tym, jak nasłuchiwać i filtrować wiadomości NDEF za pomocą detektorów zdarzeń PhoneGap-NFC i systemu wysyłania tagów Androida.

Odsłuchiwacze zdarzeń PhoneGap-NFC

Do tej pory nie określiłeś, w jaki sposób Android powinien radzić sobie z tagami i używałeś tylko jednego detektora zdarzeń, `tagDiscoveredListener`. Ten nasłuchiwacz zdarzeń korzysta z systemu wysyłania na pierwszym planie, aby

powiedzieć Androidowi, że chcesz, aby Twoja aplikacja odbierała wszystkie wiadomości związane z NFC, a nie kierowała je do innych aplikacji. Działa to dobrze, gdy chcesz, aby wszystko przechodziło przez twoją aplikację i prawdopodobnie już zorientowałeś się, że niektóre inne aplikacje, które widziałeś, takie jak TagWriter i TagInfo, również używają systemu wysyłania na pierwszym planie w ten sposób. Gdy aplikacja jest na pierwszym planie, otrzymuje priorytet

powiadomienia o wszystkich zdarzeniach, NFC i innych. Oznacza to, że to Ty decydujesz, czego chcesz słuchać, a co ignorować.

Wtyczka PhoneGap-NFC oferuje cztery różne detektory zdarzeń związanych z NFC:

Znacznik wykrył słuchacza
: Nasłuchuje wszystkich tagów, które są kompatybilne ze sprzętem czytnika. Jest to najbardziej ogólny odbiornik.

Słuchacz formatu NDEF
: Nasłuchuje wszystkich kompatybilnych tagów, które mogą być sformatowane do odbierania komunikatów NDEF.

Słuchacz NDEF
: Nasłuchuje znaczników zawierających komunikaty NDEF. Jeśli odebrany zostanie prawidłowy komunikat NDEF, ten słuchacz wygeneruje zdarzenie.

Odbiornik typu MIME
: Nasłuchuje tylko tych wiadomości NDEF, które zawierają typ MIME. Jest to najbardziej specyficzny ze wszystkich detektorów zdarzeń i często najbardziej przydatny. Listenera typu MIME można użyć do filtrowania wiadomości zawierających określony typ MIME. Następnie zignoruje wiadomości zawierające inne typy MIME. Można go również użyć do filtrowania wiadomości niezawierających żadnego typu. Nie można jednak zarejestrować dwóch listenerów typu MIME dla różnych typów.

Typy nośników MIME w NDEF

Słowo *typ ma* wiele znaczeń, gdy mówimy o NFC. Jeśli nadal masz wątpliwości, zapoznaj się z sekcjami "Format nazwy typu" na stronie 51 i "Typ ładunku" na stronie 51, które omawiają je szczegółowo. Znaczna część tego rozdziału koncentruje się na jednym konkretnym typie - typie nośnika MIME. *MIME* (pierwotnie *Multipurpose Internet Mail Extensions*) to standard internetowy pierwotnie zaprojektowany w celu rozszerzenia poczty elektronicznej o obsługę różnych typów mediów nietekstowych. Pełna lista typów MIME znajduje się w dokumencie Internet Assigned Numbers Authority (IANA) dotyczącym typów MIME. Autorzy specyfikacji NFC Forum zdali sobie sprawę, że nie muszą ponownie wymyślać protokołu typu nośnika, ponieważ MIME już wykonał całkiem dobrą robotę.

W przypadku rekordów NDEF z typem nośnika TNF MIME, zazwyczaj używa się jednego z wcześniej istniejących typów MIME, takich jak `text/plain`, jako typu rekordu do opisania zawartości ładunku. Specyfikacja NDEF mówi: "Używanie niezarejestrowanych typów mediów jest zabronione". Mimo to robimy to. Dlaczego? Typy MIME to łatwy i wygodny sposób na odróżnienie tagów NFC od innych tagów. Na przykład podczas korzystania ze świateł Hue w rozdziale 6 definiujemy typ MIME `text/hue`. Widziałeś to również w "NDEF w praktyce" na stronie 56, kiedy NFC Task Launcher używał typu o nazwie `x/nfctl`. Nie jest to typ, który można znaleźć w ogólnej specyfikacji MIME.

Android udostępnia filtry intencji (patrz "Android's Tag Dispatch System" na stronie 89 w dalszej części tego rozdziału), które pozwalają aplikacji zarejestrować się w celu nasłuchiwania tagów NFC, na których jej zależy

Informacje. Gdy system operacyjny napotka tag NFC, w którym zarejestrowana jest aplikacja
aby nasłuchiwać, uruchomi aplikację i dołączy dane z tagu NFC. NFC Task Launcher jest prawdopodobnie zarejestrowany do nasłuchiwania typu MIME `x/nfctl`. Zrobisz coś podobnego, gdy przejdziesz do rozdziału 6.

Możesz zarejestrować wiele detektorów w tej samej aplikacji, a Android wygeneruje tylko najbardziej specyficzne zdarzenie, jeśli dany tag spełnia wymagania więcej niż jednego detektora. Na przykład tag z wiadomością w postaci zwykłego tekstu może potencjalnie wywołać wszystkie te detektory. Jest to czytelny tag, jest formatowalny przez NDEF, jest formatowany przez NDEF i ma uzasadniony typ MIME (`text/plain`). Ponieważ jednak typ MIME jest najbardziej specyficznym nasłuchiwaczem, to właśnie to zdarzenie zostanie wygenerowane.

Można użyć wielu kaskadowych listenerów. Na przykład, jeśli wiesz, że większość twoich tagów to zwykłe tagi tekstowe, możesz mieć określony listener typu MIME, który je przechwytuje, a następnie użyć listenera NDEF do przechwytywania wszystkich innych typów NFC oraz listenera formatowalnego NDEF do wyszukiwania pustych tagów i monitowania użytkownika o umieszczenie na nich czegoś.

Następna aplikacja pokazuje to w akcji. W tej aplikacji zaimplementujesz wszystkie cztery nasłuchiwacze i użyjesz różnych tagów do wyzwalania każdego z nich.

Aplikacja czytnika NDEF

Do tego projektu potrzebne są następujące elementy:

- Telefon z systemem Android obsługujący technologię NFC
- Co najmniej cztery tagi NFC (aby uzyskać najlepsze wyniki na wielu urządzeniach, trzymaj się typów NFC Forum; więcej informacji na temat tego, które tagi współpracują z którymi urządzeniami, można znaleźć w sekcji "Dopasowanie typu urządzenia do tagu" na stronie 19).

Oczekujemy również, że przeszedłeś przez rozdział 3 i zainstalowałeś całe oprogramowanie potrzebne do wykonania przykładów z tego rozdziału.

Zacznij od utworzenia nowego projektu za pomocą polecenia `cordova create`:

```
$ cordova create ~/NdefReader com.example.ndefreader NdefReader ❶
$ cd ~/NdefReader ❷
$ cordova platform add android
$ cordova plugin add https://github.com/chariotsolutions/phonegap-nfc
```

❶ Użytkownicy Windows powinni wpisać `%userprofile%\NdefReader` zamiast `~/Ndef Reader`.

❷ Użytkownicy systemu Windows powinni wpisać `/d %userprofile%\NdefReader`

zamiast

```
~/NdefReader.
```

Możesz skopiować *index.html* i *index.js* z aplikacji NFC Reader jako bazę dla tego projektu:

```
$ cp ~/NfcReader/www/index.html ~/NdefReader/www/.
$ cp ~/NfcReader/www/js/index.js ~/NdefReader/www/js/.
```

Plik *index.html* dla tego projektu powinien wyglądać następująco:

```
<!DOCTYPE html>

<html>
   <head>
      <title>NDEF Reader</title>
   </head>
   <body>
      <div class="app">.
         <p>NDEF Reader</p>
         <div id="messageDiv"></div>
      </div>
      <script type="text/javascript" src="cordova.js"></script>.
      <script type="text/javascript" src="js/index.js"></script>.
      <script type="text/javascript">
       app.initialize();
     </script>
    </body>
</html>
```

Słuchanie wielu zdarzeń

Plik *index.js* zaczyna się od tej samej bazy, co aplikacja NFC Reader. Funkcje `initialize()` i `bindEvents()` pozostaną takie same, podobnie jak funkcje `display()` i `clear()`. Funkcja `onDeviceReady()` rozpocznie się tak samo, ale po `addTagDiscoveredListener` zostaną dodane trzy nowe nasłuchiwacze. Nowa funkcja będzie wyglądać następująco:

```
/*
   uruchamia się, gdy urządzenie jest gotowe do interakcji z użytkownikiem:
*/
   onDeviceReady: function() {

      nfc.addTagDiscoveredListener(
         app.onNonNdef,          // tag został pomyślnie zeskanowany
         function (status) {     // słuchacz został pomyślnie zainicjowany
            app.display("Nasłuchiwanie tagów NFC.");
         },
         function (error) {      // słuchacz nie został zainicjowany
            app.display("Nie udało się zainicjować czytnika NFC "
               + JSON.stringify(error));
         }
      );

      nfc.addNdefFormatableListener(
         app.onNonNdef,          // tag został pomyślnie zeskanowany
```

```
        function (status) {      // słuchacz został pomyślnie zainicjowany
            app.display("Listening for NDEF Formatable tags.");
        },
        function (error) {       // słuchacz nie został zainicjowany
            app.display("Nie udało się zainicjować czytnika NFC "
                + JSON.stringify(error));
        }
    );

    nfc.addNdefListener(
        app.onNfc,               // tag został pomyślnie zeskanowany
        function (status) {      // słuchacz został pomyślnie zainicjowany
            app.display("Nasłuchiwanie komunikatów NDEF.");
        },
        function (error) {       // słuchacz nie został zainicjowany
            app.display("Nie udało się zainicjować czytnika NFC "
                + JSON.stringify(error));
        }
    );

    nfc.addMimeTypeListener(
        "text/plain",
        app.onNfc,               // tag został pomyślnie zeskanowany
        function (status) {      // słuchacz został pomyślnie zainicjowany
            app.display("Listening for plain text MIME Types.");
        },
        function (error) {       // słuchacz nie został zainicjowany
            app.display("Nie udało się zainicjować czytnika NFC "
                + JSON.stringify(error));
        }
    );

    app.display("Dotknij tagu, aby odczytać dane.");
},
```

Jak widać, struktura tych detektorów jest w większości taka sama. Jedyną zmianą jest komunikat, który wyświetlają po pomyślnym zarejestrowaniu oraz funkcja zwrotna, którą wywołują. Te, które prawdopodobnie zwrócą komunikat NDEF, wywołują `onNfc()`, a te, które tego nie zrobią, wywołują `onNonNdef()`.

Jeśli odczytany tag nie jest w żaden sposób kompatybilny z NDEF, listener nie zwróci tagu. Na przykład można odczytać tagi RFID 13,56 MHz, które nie są nawet tagami Mifare (Philips ICODE lub Texas Instruments Tag-it); czytnik może odczytać jego UID, ale nic więcej. Jeśli korzystasz z urządzenia takiego jak Nexus 4, które nie może odczytywać tagów Mifare Classic, będziesz w stanie odczytać tylko identyfikator UID, ale żadnych danych tagu. Należy obsługiwać te tagi w inny sposób. Obsługa `onNonNdef()` zajmuje się tym przypadkiem.

Nowa funkcja `onNfc()` wywoła `clear()` w celu wyczyszczenia komunikatu div, a następnie wyświetli typ zdarzenia, które go wywołało oraz szczegóły dotyczące tagu. Funkcja `onNonNdef()` będzie

odczytuje UID tagu i typy techniczne z tagu. Następnie dodaj funkcję `showTag()`, aby wyświetlić szczegóły tagu z `onNfc()`:

```
    /*
       Przetwarzanie danych znacznika NDEF ze zdarzenia nfcEvent
    */
    onNfc: function(nfcEvent) {
       app.clear();              // wyczyszczenie komunikatu div
       // wyświetla typ zdarzenia:
       app.display(" Typ zdarzenia: " + nfcEvent.type);
       app.showTag(nfcEvent.tag); // wyświetla szczegóły
       tagu
    },

    /*
       Przetwarzanie danych tagów innych niż NDEF z
       nfcEvent Obejmuje to
         * Tagi NFC inne niż NDEF
         * Znaczniki formatowalne NDEF
         * Tagi Mifare Classic na Nexus 4, Samsung S4
        (ponieważ Broadcom nie obsługuje Mifare Classic)
    */
    onNonNdef: function(nfcEvent) {
       app.clear();              // wyczyszczenie komunikatu div
       // wyświetlenie typu zdarzenia:
       app.display("Typ zdarzenia: " +
       nfcEvent.type); var tag = nfcEvent.tag;
       app.display("Identyfikator tagu: " + nfc.bytesToHexString(tag.id));
       app.display("Typy techniczne: ");
       for (var i = 0; i < tag.techTypes.length; i++) {
          app.display(" * " + tag.techTypes[i]);
       }
    },

/*
   zapisuje @tag do elementu div wiadomości:
*/

    showTag: function(tag) {
       // wyświetl właściwości tagu:
       app.display("Identyfikator tagu: " + nfc.bytesToHexString(tag.id));
       app.display("Typ tagu: " + tag.type);
       app.display("Max Size: " + tag.maxSize + " bytes");
       app.display("Is Writable: " + tag.isWritable);
       app.display("Can Make Read Only: " + tag.canMakeReadOnly);
     },
```

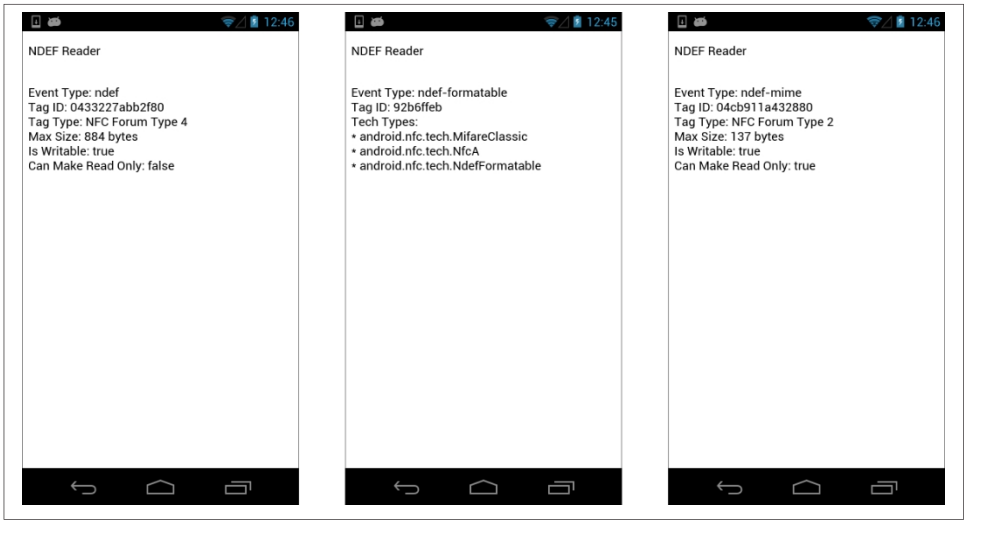
To wystarczy do uruchomienia aplikacji. Zapisz wszystkie pliki i uruchom aplikację jak zwykle:

```
$ cordova run
```

Wypróbuj tę aplikację z tagami zameldowania Foursquare, które utworzyłeś w rozdziale 4. Następnie spróbuj utworzyć nowy tag z wiadomością tekstową. Można to zrobić za

pomocą aplikacji NXP TagWriter. Stuknij "Utwórz,

Zapisz i przechowuj", następnie "Nowy", "Zwykły tekst", a następnie wprowadź wiadomość tekstową. Na koniec stuknij "Dalej", a następnie stuknij nowy tag z tyłu urządzenia, aby go zapisać. Następnie ponownie otwórz aplikację NDEF Reader, którą właśnie napisałeś i odczytaj ten tag. Powinien on zostać odczytany jako zdarzenie ndef-mime, w przeciwieństwie do pozostałych. Rysunek 5-1 pokazuje kilka różnych wyników, które powinieneś uzyskać z tej aplikacji do tej pory.



Rysunek 5-1. Wyniki z aplikacji NDEF Reader: ogólny znacznik sformatowany w NDEF; pusty, ale formatowalny w NDEF znacznik; oraz znacznik tekstowy z typem MIME "text/plain".

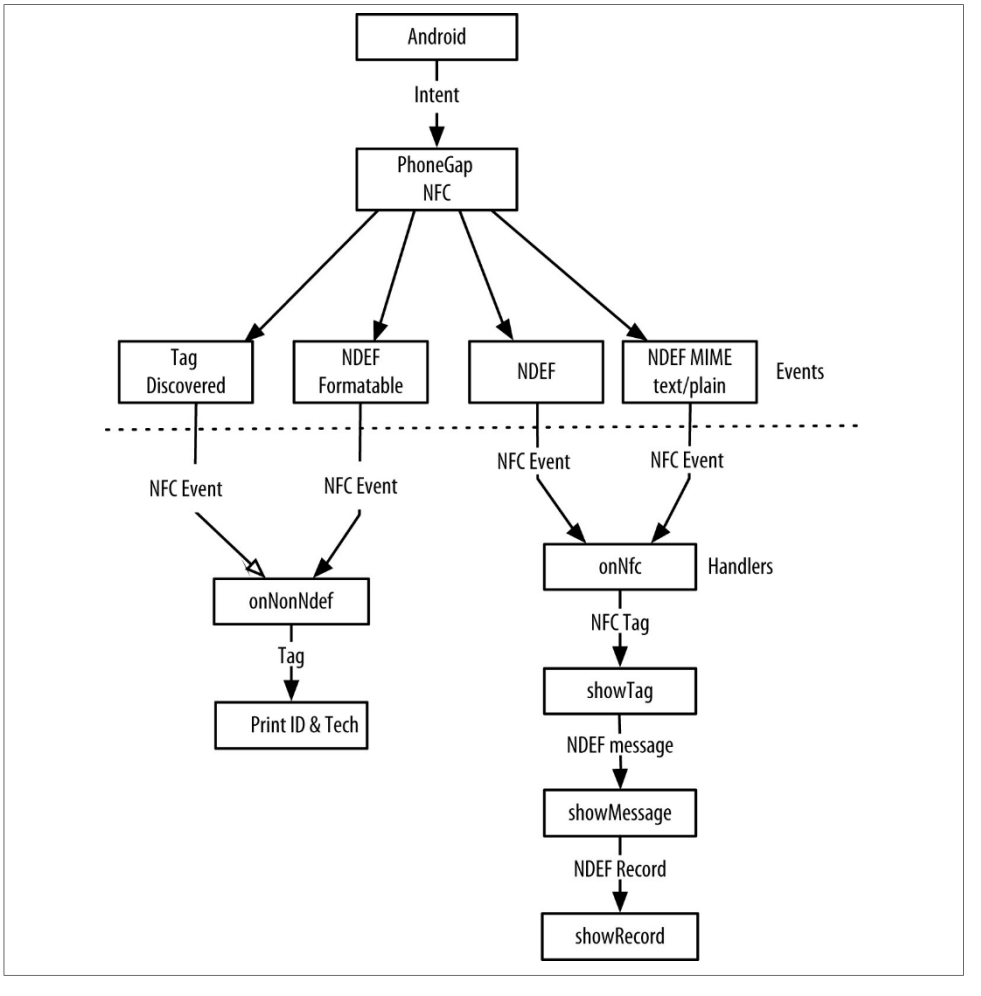
Prawdopodobnie nie otrzymałeś żadnych nowych zdarzeń typu "tag". Dzieje się tak, ponieważ jest on nadpisywany przez każdy inny detektor zdarzeń. Rzadko można natknąć się na tag, który jest kompatybilny z czytnikiem NFC, ale nie jest przynajmniej formatem NDEF. Niektóre tagi RFID inne niż Mifare pojawią się w ten sposób (w tym tagi Texas Instrument Tag-it HF i jedna z kart kredytowych Dona), ale niewiele innych. Jeśli jednak zdarzy ci się pisać aplikacje PhoneGap-NFC dla BlackBerry 7, zobaczysz, że wszystkie kwalifikujące się nasłuchiwacze zostaną uruchomione, gdy dany tag znajdzie się w zasięgu, więc będziesz musiał dostosować swój kod do użytku międzyplatformowego, jeśli chcesz, aby użytkownicy BlackBerry uzyskali takie same wyniki, jak użytkownicy Androida. Więcej informacji na temat różnic międzyplatformowych można znaleźć w pliku README wtyczki PhoneGap-NFC.

Odczytywanie komunikatów NDEF

Teraz, gdy masz już aplikację, która może odczytywać dowolny typ kompatybilnego tagu, możesz równie dobrze rozszerzyć ją, aby podawała szczegóły komunikatów NDEF na każdym tagu, który odczytuje. Należy pamiętać, że komunikaty NDEF są

tworzone z rekordów NDEF, a gdy rekord jest inteligentnym plakatem, zawartość rekordu jest sama w sobie komunikatem NDEF. Aby sobie z tym poradzić, potrzebna będzie funkcja `showMessage()` wywoływana przez `showTag()` oraz funkcja `showRecord()`

która jest wywoływana przez `showMessage()`. Jeśli rekord jest inteligentnym plakatem, należy ponownie wywołać funkcję showMessage(). Tutaj rekurencja jest twoim przyjacielem, jak zobaczysz. Rysunek 5-2 pokazuje przepływ całego programu.



Rysunek 5-2. Przepływ programu dla aplikacji NDEF Reader

Zacznij od dodania instrukcji `if` do funkcji `showTag()`, która wysyła komunikat NDEF na tagu do funkcji `showMessage()`:

```
showTag: function(tag) {
    // wyświetl właściwości tagu:
    app.display("Identyfikator tagu: " + nfc.bytesToHexString(tag.id));
    app.display("Typ tagu: " + tag.type);
    app.display("Max Size: " + tag.maxSize + " bytes");
```

```
        app.display("Is Writable: " + tag.isWritable);
        app.display("Can Make Read Only: " + tag.canMakeReadOnly);

        // jeśli na tagu znajduje się komunikat NDEF, wyświetl go:
        var thisMessage = tag.ndefMessage;
        if (thisMessage !== null) {
           // pobiera i wyświetla liczbę rekordów NDEF:
           app.display("Tag ma komunikat NDEF o długości " + thisMessage.length
              + " record" + (thisMessage.length == 1 ? ".": "s."));

           app.display("Treść wiadomości: ");
           app.showMessage(thisMessage);
        }
     },
```

Następnie dodaj funkcję `showMessage()`, aby wyświetlić wiadomość. Wiadomość jest po prostu tablicą rekordów, więc iteruj po niej, aby wyświetlić każdy rekord za pomocą drugiej funkcji, `showRecord()`:

```
     /*
        iteruje po rekordach w komunikacie NDEF, aby je wyświetlić:
     */
        showMessage: function(message) {
           for (var thisRecord in message) {
              // pobierz następny rekord w tablicy
              wiadomości: var record =
              message[thisRecord];
              app.showRecord(record);          // pokaż
              to
           }
        },
     /*
        zapisuje @record do elementu div wiadomości:
     */
        showRecord: function(record) {
           // wyświetla TNF, typ i ID:
           app.display(" ");
           app.display("TNF: " + record.tnf);
           app.display("Type: " + nfc.bytesToString(record.type));
           app.display("ID: " + nfc.bytesToString(record.id));

           // jeśli ładunkiem jest Smart Poster, jest to wiadomość NDEF.
           // odczytać go i wyświetlić (rekurencja jest tutaj twoim przyjacielem):
           if (nfc.bytesToString(record.type) == "Sp") {
              var ndefMessage = ndef.decodeMessage(record.payload);
              app.showMessage(ndefMessage);

           // jeśli ładunek nie jest inteligentnym plakatem, wyświetl go:
           } else {
              app.display("Payload: " + nfc.bytesToString(record.payload));
           }
        }
     };      // koniec aplikacji
```