Spis treści

Wprowadzenie

Tempo rozwoju systemu Android jest szalone. Na przestrzeni zaledwie kilku lat Android stał się stabilną platformą, dorównując swojemu głównemu konkurentowi, iOS. W czasie, gdy ta książka była pisana, najnowszą wersją systemu Android była wersja 4.1 (alias Jelly Bean). System Android 4.1 działa zarówno na smartfonach, jak i tabletach, więc staje się naturalnym wyborem dla wielu deweloperów.

Ta książka narodziła się w wyniku wielu frustracji, jakie miałem przy opracowywaniu aplikacji w systemie Android. Często zdarza się, że potrzebujemy tylko szybkiej migawki pokazującej, jak zrobić dane zadanie, a fragment kodu byłby idealnym rozwiązaniem. Jednak odwołanie się do oficjalnej dokumentacji systemu Android często powoduje więcej nieporozumień niż pomocy, ponieważ przykłady kodu nie zawsze są kompletne. Tak więc celem tej książki jest wypełnienie luki poprzez podanie niezależnych przykładów, które można szybko ogarnąć i je rozwinąć.

Każdy przepis dotyczy problemu, który możemy napotkać w swojej codziennej pracy dewelopera systemu Android – niezależnie od tego, czy jest tak drobna kwestia, tak jak korzystanie z widoku za pomocą opcji Button (Przycisk), bądź tak zawiła, jak implementowanie aplikacji Master-Detail z wykorzystaniem fragmentów. Tę książkę można czytać od pierwszego przepisu aż do ostatniego lub bezpośrednio przejść do przepisów, które są dla nas najbardziej interesujące.

Dla kogo jest ta książka

Książka jest przeznaczona dla programistów pracujących w systemie Android, którzy mają już jakąś podstawową wiedzę o tworzeniu aplikacji w tym systemie. Zakłada się, że Czytelnicy wiedzą, jak tworzyć projekt w systemie Android za pomocą Eclipse i znają już strukturę projektu w tym systemie.

Wszystkie przykłady kodu w tej książce zostały napisane i przetestowane przy użyciu Android 4.1 SDK, razem z Eclipse (wydanie Juno) oraz wtyczką ADT 20.0.3. Wszystkie projekty mogą działać na urządzeniach z systemem Android począwszy od wersji 2.2 tego systemu. W szczególności, we wszystkich projektach wykorzystano pakiet Android Support , który jest domyślnie włączony do projektów Android 4.1. Dzięki zastosowaniu Android Support Package nasze aplikacje mogą korzystać z nowych funkcji wprowadzonych w wersji 3.0 systemu Android (takich jak fragmenty), a jednocześnie działają na urządzeniach ze starszymi wersjami systemu.

> **UWAGA** *Mimo dołożenia wszelkich starań, aby wszystkie stosowane w przykładach narzędzia były najnowsze, zawsze może się zdarzyć, że do momentu lektury tej książki pojawią się ich nowe wersje. W takim przypadku niektóre z instrukcji i/lub mogą się nieznacznie różnić. Jednak ze wszelkimi zmianami powinniśmy dać sobie radę.*

Co zawiera ta książka

Ta książka obejmuje wszystkie najważniejsze obszary programowania w systemie Android przy wykorzystaniu Android 4.1 SDK. Jest podzielona na 11 rozdziałów.

- Rozdział 1: **Podstawy systemu Android** zawiera tematykę podstawową, taką jak sposoby łączenia czynności, przenoszenie danych pomiędzy czynnościami, wysyłanie i odbieranie rozgłoszeń, wywoływanie wbudowanych aplikacji i wiele więcej.
- Rozdział 2: **Stosowanie widoków do projektowania interfejsu użytkownika** zawiera objaśnienie sposobu korzystania z różnych widoków do budowania interfejsu użytkownika w aplikacjach systemu Android. Zawiera również opis różnych typów układów obsługiwanych w systemie Android do rozmieszczania tych widoków, w tym LinearLayout, RelativeLayout, FrameLayout. Nauczymy się także wyświetlania menu kontekstowych i menu opcji.
- Rozdział 3: **Wyświetlanie list elementów i obrazów** zawiera opis sposobu korzystania z ListView i Spinner oraz sposobu ich dostosowania w celu wyświetlania listy elementów. Pokazano w nim również, jak korzystać z fragmentów, aby tworzyć aplikacje Master-Detail.
- Rozdział 4: **Telefonia** obejmuje tematykę związaną z telefonem w naszym urządzeniu z systemem Android, w tym sposób blokowania połączeń wychodzących, automatyczne odpowiedzi na połączenia przychodzące, włączanie Bluetooth i inne.
- Rozdział 5: **Obsługa wiadomości** zawiera opis, jak wysyłać i odbierać wiadomości SMS na swoim telefonie z systemem Android. Dowiemy się także, jak śledzić wiadomości SMS wysyłane przez użytkowników.
- Rozdział 6: **Programowanie w sieci** obejmuje tematykę związaną z łączeniem się aplikacji systemu Android ze światem zewnętrznym. Dowiemy się, jak korzystać z usług XML oraz JSON sieci Web, programowania gniazd i komunikacji Bluetooth.

- Rozdział 7: **Korzystanie z Google Maps** obejmuje tematykę dotyczącą sposobu wyświetlania Google Maps w naszej aplikacji Android oraz m.in. wykonywania odwrotnego geokodowania.
- Rozdział 8: **Usługi danych oparte na położeniu** dotyczy najważniejszych technik, które trzeba znać, aby tworzyć usługi oparte na lokalizacji. Dowiemy się także, jak implementować rejestrowanie danych lokalizacji.
- Rozdział 9: **Uzyskiwanie dostępu do sprzętu** zawiera opis sposobów uzyskiwania z aplikacji systemu Android dostępu do wielu funkcji sprzętowych. Obejmuje przepisy pokazujące, jak zrobić zdjęcie za pomocą wbudowanego aparatu fotograficznego, jak włączyć i wyłączyć GPS, a także jak włączyć lampę błyskową w urządzeniu.
- Rozdział 10: **Przechowywanie danych** obejmuje kilka metod utrzymywania danych, w tym pamięć wewnętrzną, pamięć zewnętrzną, bazę danych i inne.
- Rozdział 11: **Wdrażanie aplikacji systemu Android** zawiera opis różnych metod wdrażania aplikacji systemu Android, takich jak za pomocą karty SD, serwera Web lub poczty e-mail.

Jak zorganizowana jest ta książka

Rozdziały książki są podzielone na tematy główne, a każdy rozdział zawiera wiele „przepisów", w bardziej szczegółowy sposób odnoszących się do określonych podtematów. Zamiast tworzenia projektu krok po kroku, a następnie objaśnienia, jak działa kod, w książce zaprezentowano kluczowe elementy każdego przepisu – kluczowe punkty, które trzeba zrozumieć, aby spełnić wymagania lub rozwiązać problem (bądź wykonać typowe zadanie) w programowaniu w systemie Android. Każdy przepis obejmuje główne pojęcia, które należy zrozumieć, bez żadnego zbędnego kodu, który mógłby skomplikować przykład. Przy takim podejściu najprościej jest skopiować i wkleić kod do własnego projektu, aby następnie poprawić go, dostosowując do swoich własnych celów. Moim zdaniem jest to najlepszy sposób nauczenia się programowania w systemie Android.

W tym celu każdy rozdział ma bardzo określoną strukturę. Każdy przepis ma swój numer i tytuł. Przepis zaczyna się od listy składników potrzebnych do skutecznego rozwiązania zadania, jak na następnej stronie:

Przepis 0.0 Wymagania dla przepisu

Wersje systemu Android

W każdym przepisie wykorzystano interfejsy API z Android SDK. W punkcie „Wersje systemu Android" podano wersję (numer poziomu), z której pochodzą interfejsy API. Możemy na przykład zobaczyć „Poziom 1 i wyższe". To wskazuje, że interfejsy API stosowane w tym przepisie są dostępne w systemie Android 1 poziomu 1 (tj. wersji 1.0) i wersjach wyższych.

Uprawnienia

W tym punkcie pokazane są uprawnienia, które trzeba dodać do aplikacji, a dokładnie do pliku `AndroidManifest.xml`, aby użyć interfejsów API opisanych w danym przepisie. Należy sprawdzić, czy uprawnienie zostało dodane; jeśli o tym zapomnimy, zwykle spowoduje to błąd aplikacji podczas jej wykonywania.

Kod źródłowy do pobrania z witryny Wrox.com

W tym punkcie podana jest nazwa pliku ZIP, który można pobrać z pomocniczej witryny Web dla tej książki – z Wrox.com. Plik ZIP zawiera pełny projekt użyty do ilustracji pomysłu zawartego w tym przepisie. Jeśli chcemy szybko umieścić kod przepisu w swoim własnym projekcie, pobranie kodu źródłowego jest najszybszą opcją.

Następnie wyjaśniono główny cel danego przepisu. Dalej następuje rozwiązanie. Czasami rozwiązanie jest krótkie i przyjemne, a czasami jest bardziej złożone i wymaga wielu kroków.

Co jest potrzebne, aby korzystać z tej książki

Większość przykładów w tej książce działa na emulatorze systemu Android, który jest częścią Android SDK. Aby jednak wynieść jak najwięcej z tej książki, zaleca się skorzystanie z rzeczywistego urządzenia z systemem Android (chociaż nie jest to absolutnie konieczne).

Ponadto każdy przepis zaczyna się od listy wymagań właściwych dla tego przepisu (wersje, uprawnienia oraz kod źródłowy), jak to wcześniej omówiono.

Uwaga dotycząca włączania uprawnień do kodu

Aby przepisy stały się zwięzłe i proste do śledzenia, w tej książce założono, że Czytelnicy wiedzą, jak dodawać uprawnienia do swoich aplikacji. Na przykład zamiast pełnego listingu pliku `AndroidManifest.xml` z wyróżnieniem uprawnień, które trzeba dodać w następujący sposób:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.http"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.
                LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

przepis wskaże po prostu, że trzeba dodać następujące uprawnienia:

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

Jeśli powyższy krok nie jest bezpośrednio i wyraźnie opisany, trzeba zawsze sprawdzić sekcję Uprawnienia na początku każdego przepisu i dodać uprawnienia do pliku `AndroidManifest.xml`.

Konwencje

Aby pomóc w optymalnym korzystaniu z tekstu i śledzeniu działań, w całej książce zastosowano kilka konwencji:

- Nowe pojęcia i ważne słowa są przy pierwszym wystąpieniu *wyróżnione* kursywą.
- Kombinacje klawiszy są oznaczane tak: Ctrl+R.
- Nazwy plików, kod w tekście oraz adresy URL są oznaczane tak: `persistence.properties`.
- Kody przedstawiane są na dwa różne sposoby:

```
W większości przykładów kodu stosowany jest typ czcionki o stałej szerokości bez
wyróżnienia.
Do wyróżnienia kodu, który jest szczególnie ważny w przestawionym przykładzie,
stosowane jest pogrubienie.
```

> **UWAGA** *Uwagi, wskazówki, porady i komentarze do aktualnego omówienia wyglądają tak jak niniejsza uwaga.*

Kod źródłowy

Podczas pracy nad kolejnymi przykładami z tej książki, można wybrać ręczne wpisywanie całego kodu, albo skorzystać z plików kodu źródłowego, które towarzyszą książce. Cały kod źródłowy użyty w tej książce jest dostępny do pobrania na stronie *www.wrox.com*. Po wejściu na tę stronę znajdujemy po prostu tytuł książki (za pomocą pola Search lub jednej z list tytułów) i klikamy łącze Download Code (Pobierz kod) na stronie ze szczegółami książki, aby uzyskać jej cały kod źródłowy.

Po pobraniu kodu dokonujemy jego dekompresji za pomocą wybranego narzędzia. Alternatywnie przechodzimy do głównej strony pobierania kodu Wrox pod adresem *www.wrox.com/dynamic/books/download.aspx*, aby zobaczyć kod dostępny dla tej książki (i wszystkich pozostałych książek Wrox).

> **UWAGA** *Wyszukując książkę na liście, trzeba odwołać się do tytułu angielskiego, czyli* Android Application Development Cookbook: 93 Recipes for Building Winning Apps. *Ponieważ wiele książek może mieć podobne tytuły, łatwiejsze może być jej szukanie poprzez ISBN oryginalnego wydania angielskiego, którym dla tej książki jest* 978-1-118-17767-9.

Errata

Dołożyliśmy wszelkich starań, aby tekst ani kod nie zawierały błędów. Nikt jednak nie jest doskonały, a błędy się zdarzają. Będziemy wdzięczni za informacje w przypadku znalezienia w jednej z naszych książek błędu w pisowni lub błędnego fragmentu kodu. Wysyłając informację o błędzie możemy oszczędzić innemu Czytelnikowi godzin frustracji, a jednocześnie pomaga nam w dostarczeniu informacje jeszcze lepszej jakości.

Aby znaleźć stronę erraty dla tej książki, przechodzimy do strony *www.wrox.com* i znajdujemy tytuł, korzystając z okna Search lub jednej z list tytułów. Następnie, na stronie ze szczegółowymi informacjami o książce, klikamy łącze Book Errata. Na tej stronie można zobaczyć całą listę błędów, które zostały zgłoszone do tej książki i opublikowane przez redaktorów wydawnictwa Wrox. Pełna lista książek, wraz z łączami do ich errat, jest dostępna również pod adresem *www.wrox.com/misc-pages/booklist.shtml*.

Jeśli na stronie Book Errata Czytelnik nie dostrzeże „swojego" błędu, powinien przejść do *www.wrox.com/contact/techsupport.shtml* i wypełnić tam formularz, aby przesłać informacje o znalezionym błędzie. Informacje te zostaną sprawdzone, a jeśli zajdzie taka potrzeba, na stronie erraty pojawi się post, a w następnych wydaniach książki błąd zostanie skorygowany[1].

p2p.wrox.com

Aby podyskutować z autorem i innymi osobami, można dołączyć do forum P2P na stronie *p2p.wrox.com*. Forum to system oparty na sieci Web służący do publikowania wiadomości dotyczących książek wydawnictwa Wrox i związanych z nimi technologii oraz do wzajemnych kontaktów z innymi czytelnikami i użytkownikami technologii. Fora oferują funkcję subskrypcji na otrzymywanie pocztą e-mail nowych postów zamieszczanych na forum dotyczących wybranych, interesujących tematów. Na tych forach obecni są autorzy i redaktorzy wydawnictwa Wrox oraz inni eksperci w branży i czytelnicy.

Na stronie *p2p.wrox.com* można znaleźć kilka różnych forów, które pomogą nie tylko przy czytaniu tej książki, lecz także przy opracowywaniu swoich własnych aplikacji. Aby dołączyć do forum, należy wykonać następujące kroki:

1. Przejdź na stronę *p2p.wrox.com* i kliknij łącze Register (Zarejestruj się).

1 Przedstawiona strona zawiera erratę do wydania oryginalnego (angielskiego). Może być ona użyteczna w przypadku błędów dostrzeżonych w przykładach kodu. W razie dostrzeżenia błędu w polskim przekładzie prosimy o przesłanie informacji na adres *mspress@promise.pl*. Errata do polskiego wydania będzie dostępna na stronie poświęconej książce w witrynie *www.książki.promise.pl* (przyp. redakcji wydania polskiego).

2. Przeczytaj warunki korzystania i kliknij Agree (Zgadzam się).
3. Podaj wymagane informacje, aby dołączyć do forum, a także wszelkie informacje opcjonalne, jakie chcesz podać i kliknij Submit (Wyślij).
4. Otrzymasz wiadomość e-mail z informacjami, jak potwierdzić swoje konto i dokończyć proces dołączania.

> **UWAGA** *Wiadomości na forach można czytać bez dołączania do P2P, lecz publikowanie własnych postów wymaga dołączenia do forum.*

Po dołączeniu do forum można publikować nowe wiadomości i odpowiadać na posty innych użytkowników. Wiadomości można czytać w dowolnej chwili w sieci Web. Jeśli chcemy, aby nowe wiadomości z konkretnego forum były do nas przesyłane pocztą e-mail, trzeba kliknąć ikonę Subscribe to This Forum (Subskrybuj to forum) przy nazwie forum na liście forów.

Więcej informacji na temat sposobu korzystania z Wrox P2P, działania oprogramowania, a także odpowiedzi na wiele pytań specyficznych dla książek wydawnictwa Wrox można znaleźć w pytaniach FAQ P2P. Aby przeczytać FAQ, trzeba kliknąć łącze FAQ na dowolnej stronie P2P.

1

Podstawy systemu Android

W tym rozdziale poznamy podstawowe zagadnienia związane z systemem Android, które powinna znać większość programistów. Wśród nich są kwestie połączeń z innymi aplikacjami za pomocą obiektów `Intent`, komunikacja z innymi aplikacjami (lub częściami tej samej aplikacji) za pomocą odbiorników komunikatów oraz przekazywanie danych pomiędzy działaniami.

Przepis 1.1 Łączenie działań

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

Brak

Kod źródłowy do pobrania z witryny Wrox.com

Linking.zip

O ile nie piszemy programu typu Hello World, najpewniej nasza aplikacja będzie zawierała kilka potrzebnych działań, które należy połączyć ze sobą, aby utworzyć spójną aplikację. W tym przepisie pokazano różne sposoby łączenia działań z innymi działaniami w naszej aplikacji systemu Android.

Rozwiązanie

Przypuśćmy, że w naszej aplikacji występują dwa działania. W poniższym pliku `AndroidManifest.xml` pokazano dwie klasy działań, `MainActivity` i `Activity2`:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.linking"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name=".Activity2"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="net.learn2develop.Activity2" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Przyjmując, że jesteśmy aktualnie w działaniu `MainActivity`, to w celu połączenia się z działaniem `Activity2` możemy użyć następującego fragmentu kodu:

```
package net.learn2develop.linking;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

public class MainActivity extends Activity {
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //--- Połączenie z Activity2 ---
        Intent i = new Intent("net.learn2develop.Activity2");
        startActivity(i);
    }

}
```

Aby połączyć się z innym działaniem, musimy utworzyć obiekt `Intent` i ustawić jego konstruktor (podobnie jak element w pliku `AndroidManifest.xml`) na nazwę docelowego działania. Następnie wywołujemy metodę `startActivity()`, aby uruchomić to działanie.

Alternatywnie możemy utworzyć obiekt `Intent`, a następnie wywołać jego metodę `setAction()`, aby ustawić nazwę działania docelowego:

```
//--- Połączenie z Activity2 ---
Intent i = new Intent();
i.setAction("net.learn2develop.Activity2");
startActivity(i);
```

Powyższe fragmenty kodu są wygodne do wywołania działania, które znajduje się w tej samej aplikacji, a także do tego, aby inne aplikacje wywoływały nasze działanie. Jeśli chcemy wywołać działanie, które znajduje się wewnątrz naszej aplikacji, możemy je również wywołać w następujący sposób, korzystając z jego nazwy klasy:

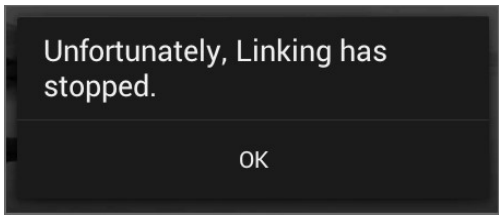
```
//--- Połączenie z Activity2---
Intent i = new Intent(this, Activity2.class);
```

Jeśli nie chcemy, aby inne działania wywoływały nasze działanie spoza naszej aplikacji, musimy po prostu usunąć element `<action>` z elementu `<intent-filter>`:

```
<activity
    android:name=".Activity2"
    android:label="@string/app_name" >
    <intent-filter>
        <!--
        <action android:name="net.learn2develop.Activity2" />
        -->
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Jeśli na danym urządzeniu nie ma działania, które próbujemy wywołać, wtedy nasza aplikacja ulegnie awarii, wyświetlając komunikat podobny do pokazanego na rysunku 1-1.

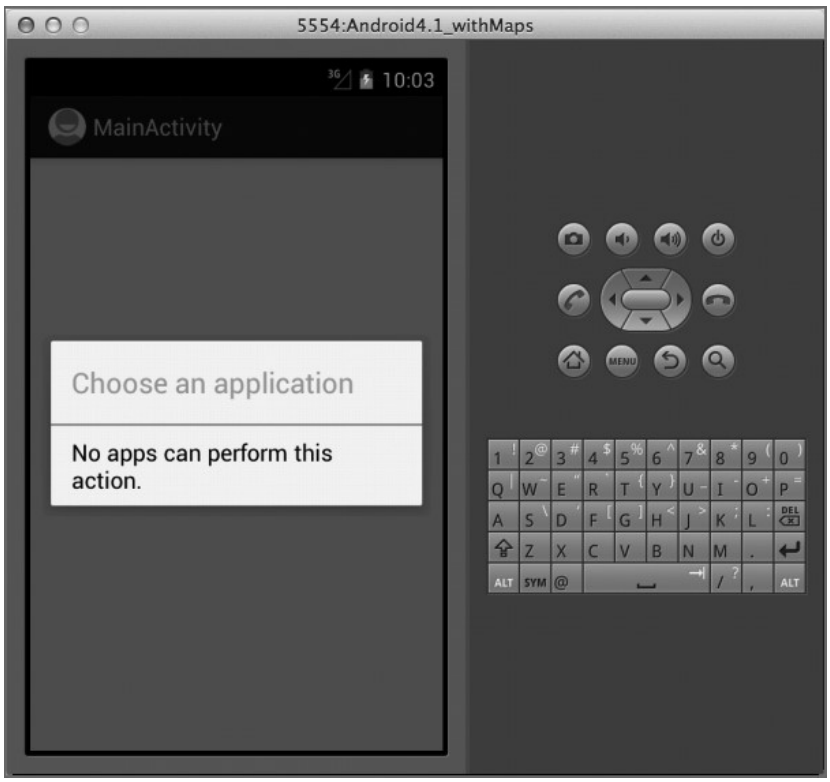
Aby nasza aplikacja nie została przerwana w sposób nagły, wywołujemy metodę `startActivity()` wraz z metodą `createChooser()`. obiektu typu `Intent`. Metoda `createChooser()` pobiera nazwę obiektu typu `Intent` oraz podany tekst, aby wyświetlić je w przypadku, gdy działania nie można znaleźć (lub gdy znaleziono więcej niż jedno działanie odpowiadające naszemu obiektowi typu `Intent`):



Rysunek 1-1

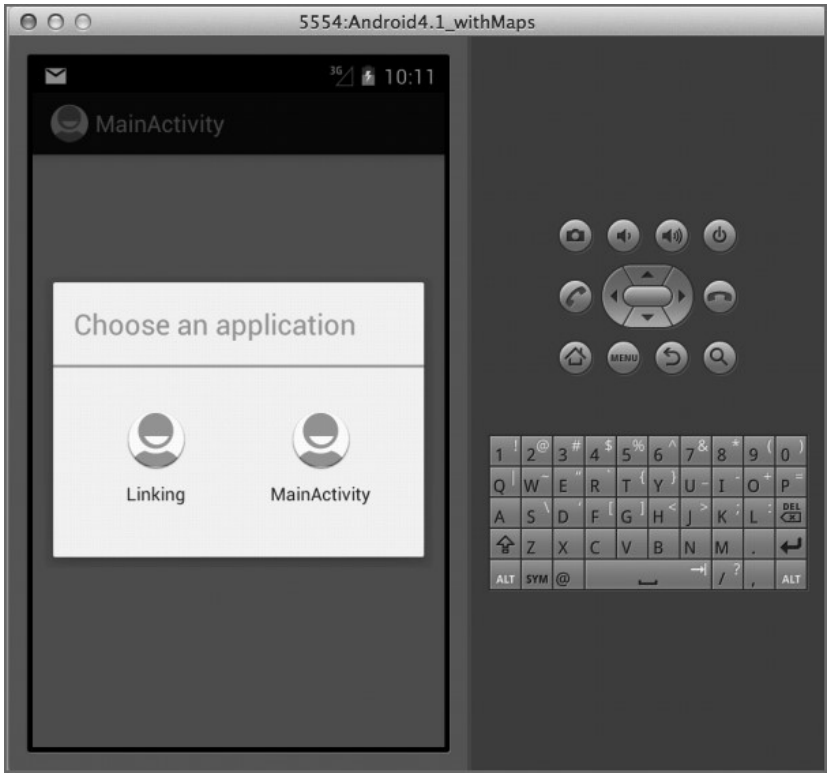
```
Intent i = new Intent("net.learn2develop.Activity2");
startActivity(Intent.createChooser(i, "Choose an application"));
```

Na rysunku 1-2 pokazano komunikat, który zostanie wyświetlony, gdy działania nie uda się znaleźć.



Rysunek 1-2

Na rysunku 1-3 pokazano komunikat, który zostanie wyświetlony, gdy program znajdzie więcej niż jedno działanie.



Rysunek 1-3

Zauważmy, że podczas korzystania z metody `createChooser()` musimy podać nazwę działania (jak na przykład `net.learn2develop.Activity2`, co pokazano w poprzednim przykładzie), które uruchamiamy, a nie nazwę jego klasy. Poniższy fragment kodu nie będzie działał:

```
//--- Ten kod nigdy nie połączy się z Activity2 ---
Intent i = new Intent(this, Activity2.class);
startActivity(Intent.createChooser(i, "Choose an application"));
```

Przepis 1.2 Przekazywanie danych między działaniami

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

Brak

Kod źródłowy do pobrania z witryny Wrox.com

PassingData.zip

W poprzednim przepisie pokazano, jak uruchomić inne działanie za pomocą metody `startActivity()`. Częstym zadaniem, które musimy wykonać podczas uruchamiania innego działania, jest przekazanie do niego danych. Na przykład możemy chcieć uruchomić inne działanie, które zbierze dane związane z użytkownikiem, więc trzeba przekazać do niego nazwę użytkownika. Gdy użytkownik zakończy zbieranie danych, należy je z kolei przekazać z powrotem do działania wywołującego. Musimy więc przekazywać dane między działaniami w obie strony. W tym przepisie pokażemy, jak to zrobić.

Rozwiązanie

Aby przekazać dane do innego działania, możemy użyć klasy `Intent`. W celu przekazania prostych typów danych można skorzystać z metody `putExtra()`, jak pokazuje poniższy fragment kodu:

```
package net.learn2develop.passingdata;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onClick(View view) {
        Intent i = new
                Intent("net.learn2develop.SecondActivity");

        //--- Użyj putExtra(), aby dodać nowe pary klucz/wartość ---
        i.putExtra("str1", "This is a string");
        i.putExtra("age1", 25);
    }
}
```

Powyższe instrukcje tworzą obiekt `Intent`, a następnie dołączają do niego dwie wartości, korzystając z metody `putExtra()`: jedną dla tekstu i jedną dla wartości całkowitej.

Dane możemy przekazać także w obiekcie `Bundle` z wykorzystaniem obiektu `Intent`. Obiekt `Bundle` przypomina obiekt słownika, gdyż można podać pary klucz/wartość. Aby przekazać obiekt `Bundle` za pomocą obiektu `Intent`, tworzymy obiekt `Bundle`, wypełniamy go, a następnie dołączamy do obiektu `Intent` za pomocą metody `putExtras()`, co pokazano w poniższym kodzie:

```
public void onClick(View view) {
    Intent i = new
            Intent("net.learn2develop.SecondActivity");

    //--- Użyj putExtra(), aby dodać nowe pary klucz/wartość ---
    i.putExtra("str1", "This is a string");
    i.putExtra("age1", 25);

    //--- Użyj putExtra(), aby dodać nowe pary klucz/wartość---
    Bundle extras = new Bundle();
    extras.putString("str2", "This is another string");
    extras.putInt("age2", 35);

    //---Dołącz obiekt Bundle do obiektu Intent---
    i.putExtras(extras);
}
```

Gdy za pomocą obiektu `Intent` uruchamiamy nowe działanie, dołączone do niego dane są przekazywane do działania docelowego. Aby wywołać inne działanie w celu odebrania od niego danych, korzystamy z metody `startActivityForResult()`:

```
public void onClick(View view) {
    Intent i = new
            Intent("net.learn2develop.SecondActivity");

    //--- Użyj putExtra(), aby dodać nowe pary klucz/wartość ---
    i.putExtra("str1", "This is a string");
    i.putExtra("age1", 25);

    //--- Użyj obiektu Bundle, aby dodać nowe pary
    // klucz/wartość ---
    Bundle extras = new Bundle();
    extras.putString("str2", "This is another string");
    extras.putInt("age2", 35);

    //--- Dołącz obiekt Bundle do obiektu Intent ---
    i.putExtras(extras);

    //--- Uruchom działanie, aby otrzymać wyniki ---
    startActivityForResult(i, 1);
}
```

Metoda `startActivityForResult()` pobiera obiekt Intent oraz kod żądania. Kod żądania to wartość całkowita, która jest większa lub równa zeru. Kod ten jest używany do zidentyfikowania działań zwrotnych, ponieważ możemy jednocześnie wywołać więcej niż jedno działania. Jeśli ustawimy kod żądania na wartość –1, to wywołanie `startActivityForResult()` jest tożsame z wywołaniem `startActivity()`. Oznacza to, że nie będziemy mogli uzyskać danych przekazanych z powrotem z działania docelowego.

W działaniu docelowym uzyskanie danych przekazanych do niego wymaga wykorzystania metody `getIntent()`, aby otrzymać instancję obiektu `Intent`, która została do niego

przekazana. Aby przekazać prosty typ danych za pomocą metody `putExtra()`, musimy użyć metody `get<typ>Extra()`, gdzie typ może być rodzaju `String`, `int`, `float` itd. Kolejny kod pokazuje, jak pobierane są dwa podstawowe typy danych:

```
package net.learn2develop.passingdata;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Toast;

public class SecondActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        //--- Pobierz przekazane dane za pomocą getStringExtra() ---
        Toast.makeText(this,getIntent().getStringExtra("str1"),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz przekazane dane za pomocą getIntExtra() ---
        Toast.makeText(this,Integer.toString(
            getIntent().getIntExtra("age1", 0)),
            Toast.LENGTH_SHORT).show();
    }
}
```

Aby uzyskać dane przekazane poprzez obiekt `Bundle`, użyjemy metody `getExtras()` dla obiektu `Intent`. Metoda `getExtras()` zwraca obiekt `Bundle`, który możemy wykorzystać do uzyskania różnych par klucz/wartość za pomocą metody `get<typ>()`, gdzie typem jest `String`, `int` i tak dalej:

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        //--- Pobierz przekazane dane za pomocą getStringExtra() ---
        Toast.makeText(this,getIntent().getStringExtra("str1"),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz przekazane dane za pomocą getIntExtra() ---
        Toast.makeText(this,Integer.toString(
            getIntent().getIntExtra("age1", 0)),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz przekazany obiekt Bundle ---
        Bundle bundle = getIntent().getExtras();

        //--- Pobierz dane za pomocą getString() ---
        Toast.makeText(this, bundle.getString("str2"),
```

```
            Toast.LENGTH_SHORT).show();

        //--- Pobierz dane za pomocą metody getInt() ---
        Toast.makeText(this,Integer.toString(bundle.getInt("age2")),
            Toast.LENGTH_SHORT).show();
    }
```

Działanie docelowe może także przekazywać dane z powrotem do działania, z którego nastąpiło wywołanie. Aby przekazać dane z powrotem, możemy utworzyć kolejny obiekt `Intent` i ustawić wartości we wcześniej opisany sposób. Możemy też skorzystać z metody `setData()`, aby przekazać obiekt `Uri` poprzez obiekt `Intent`. Aby przekazać wyniki do działania wywołującego, korzystamy z metody `setResult()` w sposób pokazany w poniższym kodzie:

```
package net.learn2develop.passingdata;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class SecondActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        //--- Pobierz przekazane dane za pomocą getStringExtra() ---
        Toast.makeText(this,getIntent().getStringExtra("str1"),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz przekazane dane za pomocą getIntExtra() ---
        Toast.makeText(this,Integer.toString(
            getIntent().getIntExtra("age1", 0)),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz przekazany obiekt Bundle ---
        Bundle bundle = getIntent().getExtras();

        //--- Pobierz dane za pomocą getString() ---
        Toast.makeText(this, bundle.getString("str2"),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz dane za pomocą metody getInt() ---
        Toast.makeText(this,Integer.toString(bundle.getInt("age2")),
            Toast.LENGTH_SHORT).show();
    }

    public void onClick(View view) {
        //--- Użyj obiektu Intent, aby zwrócić dane ---
```

```
        Intent i = new Intent();

        //--- Użyj metody putExtra(), aby zwrócić jakąś wartość ---
        i.putExtra("age3", 45);

        //---  Użyj metody setData(), aby zwrócić jakąś wartość ---
        i.setData(Uri.parse(
            "http://www.learn2develop.net"));

        //--- Ustaw wyniki jako OK oraz obiekt Intent ---
        setResult(RESULT_OK, i);

        finish();
    }
}
```

Stała `RESULT_OK` pozwala nam na wskazanie działaniu wywołującemu, czy zwracane dane powinny zostać zignorowane. Jeśli chcemy, aby działanie wywołujące zignorowało wyniki, możemy użyć stałej `RESULT_CANCELLED`. Sposób interpretacji wyników zależy od działania wywołującego, ale użycie tych dwóch stałych służy jako wskaźnik.

W głównym działaniu wywołującym implementujemy metodę `onActivityResult()`. Musimy sprawdzić, czy jest żądany kod, aby sprawdzić, czy otrzymujemy wyniki z odpowiedniego działania. Żądany kod to numer, który wcześniej przekazaliśmy do metody `startActivityForResult()`. W tym przykładzie ma on wartość 1:

```
        //--- Uruchom działanie, aby otrzymać wynik ---
        startActivityForResult(i, 1);
```

Możemy także sprawdzić kod wyniku, aby zobaczyć, czy wynikiem jest `RESULT_OK`:

```
package net.learn2develop.passingdata;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onClick(View view) {
        Intent i = new
                Intent("net.learn2develop.SecondActivity");
```

```
        //--- Użyj putExtra(), aby dodać nowe pary klucz/wartość ---
        i.putExtra("str1", "This is a string");
        i.putExtra("age1", 25);

        //--- Użyj obiektu Bundle, aby dodać nowe pary
        // klucz/wartość ---
        Bundle extras = new Bundle();
        extras.putString("str2", "This is another string");
        extras.putInt("age2", 35);

        //--- Dołącz obiekt Bundle do obiektu Intent ---
        i.putExtras(extras);

        //--- Uruchom działanie, aby otrzymać jego wyniki ---
        startActivityForResult(i, 1);
    }

    public void onActivityResult(int requestCode,
            int resultCode, Intent data)
    {
        //--- Sprawdź, czy żądany kod ma wartość 1 ---
        if (requestCode == 1) {

            //--- Jeśli wynik jest OK ---
            if (resultCode == RESULT_OK) {

                //--- Pobierz wyniki za pomocą getIntExtra() ---
                Toast.makeText(this, Integer.toString(
                        data.getIntExtra("age3", 0)),
                        Toast.LENGTH_SHORT).show();

                //--- Pobierz wyniki za pomocą getData()---
                Uri url = data.getData();
                Toast.makeText(this, url.toString(),
                        Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Aby pobrać dane wysłane za pomocą metody `setData()`, korzystamy z metody `getData()` obiektu `Intent` (przekazanego jako drugi argument metody `ActivityResult()`).

Przepis 1.3 Przekazywanie obiektów pomiędzy działaniami

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

Brak

Kod źródłowy do pobrania z witryny Wrox.com

PassingData.zip

W poprzednim przepisie zobaczyliśmy, jak przekazywać proste dane (jak teksty i wartości całkowite) pomiędzy działaniami. W tym przepisie pokazano, jak między działaniami przekazywać obiekty. Na przykład w obiekcie możemy mieć hermetyzowane informacje o kliencie (takie jak identyfikator klienta, nazwisko, firma i tak dalej), które trzeba przekazywać do innego działania w celu ich przetworzenia. Zamiast oddzielnego przekazywania poszczególnych fragmentów informacji o kliencie, łatwiej będzie przekazać cały obiekt.

Rozwiązanie

Poza przekazywaniem prostych typów danych za pomocą metod `putExtra()` i `putExtras()`, możemy także przekazywać obiekty za pomocą obiektu `Intent`. Jeśli mamy własną niestandardową klasę, musimy sprawić, aby klasa ta implementowała podstawową klasę `Serializable`. Przykładem jest poniższa klasa `MyCustomClass`:

```
package net.learn2develop.passingdata;

import java.io.Serializable;

public class MyCustomClass implements Serializable {
    private static final long serialVersionUID = 1L;
    String _name;
    String _email;

    public void setName(String name) {
        _name = name;
    }

    public String Name() {
        return _name;
    }

    public void setEmail(String email) {
```

```
        _email = email;
    }

    public String Email() {
        return _email;
    }
}
```

Aby przekazać obiekt do innego działania, korzystamy z metody `putExtra()`:

```
public void onClick(View view) {
    Intent i = new
            Intent("net.learn2develop.SecondActivity");

    //--- Użyj putExtra(), aby dodać nowe pary klucz/wartość ---
    i.putExtra("str1", "This is a string");
    i.putExtra("age1", 25);

    //--- Użyj obiektu Bundle, aby dodać nowe pary
    // klucz/wartość ---
    Bundle extras = new Bundle();
    extras.putString("str2", "This is another string");
    extras.putInt("age2", 35);

    //--- Dołącz obiekt Bundle do obiektu Intent ---
    i.putExtras(extras);

    //--- Utwórz swój własny obiekt niestandardowy ---
    MyCustomClass myObject = new MyCustomClass();
    myObject.setName("Wei-Meng Lee");
    myObject.setEmail("weimenglee@learn2develop.net");
    i.putExtra("MyObject", myObject);

    //--- Uruchom działanie, aby otrzymać wyniki ---
    startActivityForResult(i, 1);
}
```

Aby pobrać obiekt przekazany do innego działania, korzystamy z metody `getSerializableExtra()` obiektu `Intent`, przekazując mu klucz, który wcześniej ustalamy w metodzie `putExtra()`. Następnie umieszczamy wyniki otrzymane przez tę metodę w klasie `MyCustomClass` i przypisujemy je do zmiennej danego typu:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);

    //--- Pobierz przekazane dane za pomocą getStringExtra() ---
    Toast.makeText(this,getIntent().getStringExtra("str1"),
        Toast.LENGTH_SHORT).show();

    //--- Pobierz przekazane dane za pomocą getIntExtra() ---
    Toast.makeText(this,Integer.toString(
```

```
            getIntent().getIntExtra("age1", 0)),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz przekazany obiekt Bundle ---
        Bundle bundle = getIntent().getExtras();

        //--- Pobierz dane za pomocą getString() ---
        Toast.makeText(this, bundle.getString("str2"),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz dane za pomocą metody getInt() ---
        Toast.makeText(this,Integer.toString(bundle.getInt("age2")),
            Toast.LENGTH_SHORT).show();

        //--- Pobierz niestandardowy przekazany obiekt ---
        MyCustomClass obj = (MyCustomClass)
            getIntent().getSerializableExtra("MyObject");
        Toast.makeText(this, obj.Name(), Toast.LENGTH_SHORT).show();
        Toast.makeText(this, obj.Email(), Toast.LENGTH_SHORT).show();
    }
```

Przepis 1.4 Wysyłanie i odbieranie komunikatów

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

Brak

Kod źródłowy do pobrania z witryny Wrox.com

UsingBroadcastReceiver.zip

Obsługa komunikatów w systemie Android umożliwia nam wysyłanie komunikatów do innych części naszej aplikacji (lub innych aplikacji), abyśmy mogli informować ją o jakichś zdarzeniach. W tym przepisie nauczymy się, jak tworzyć odbiornik komunikatów do ich nasłuchiwania, a także jak wysyłać komunikaty do innych aplikacji.

Rozwiązanie

Są dwa sposoby tworzenia odbiornika komunikatów: programowy za pomocą kodu oraz deklaratywny za pośrednictwem pliku `AndroidManifest.xml`. W poniższym punkcie zostanie omówione każde możliwe rozwiązanie.

Programowe rejestrowanie odbiornika komunikatów

Rozważmy poniższe działanie:

```
package net.learn2develop.usingbroadcastreceiver;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends Activity {
    MyBroadcastReceiver myReceiver;
    IntentFilter intentFilter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        myReceiver = new MyBroadcastReceiver();
        intentFilter = new IntentFilter("MY_SPECIFIC_ACTION");
    }

    @Override
    public void onResume() {
        super.onResume();
        //--- Zarejestruj odbiornik ---
        registerReceiver(myReceiver, intentFilter);
    }

    @Override
    public void onPause() {
        super.onPause();
        //--- Wyrejestruj odbiornik ---
        unregisterReceiver(myReceiver);
    }

    public void onClick(View view) {
        Intent i = new Intent("MY_SPECIFIC_ACTION");
        i.putExtra("key", "some value from intent");
        sendBroadcast(i);
    }

    public class MyBroadcastReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent i) {
            Toast.makeText(context,
                    "Received broadcast in MyBroadcastReceiver, " +
                    " value received: " + i.getStringExtra("key"),
```

```
                    Toast.LENGTH_LONG).show();
            }
        }
    }
```

Powyższy fragment kodu pokazuje wewnętrzną klasę `MyBroadcastReceiver` rozszerzającą klasę podstawową `BroadcastReceiver`. W tej klasie musimy przesłonić metodę `onReceive()` tak, aby po otrzymaniu komunikatu można było zrealizować działanie, które chcemy wykonać. Aby dane zostały przekazane do odbiornika, możemy wykorzystać obiekt `Intent` w drugim argumencie metody `onReceive()`.

Aby użyć tej klasy, musimy utworzyć jej instancję, a także obiekt `IntentFilter`:

```
myReceiver = new MyBroadcastReceiver();
intentFilter = new IntentFilter("MY_SPECIFIC_ACTION");
```

W konstruktorze obiektu `IntentFilter` podajemy działanie zdefiniowane przez użytkownika i wykorzystujemy własny łańcuch testowy do zdefiniowania działania.

Aby zarejestrować obiekt `BroadcastReceiver`, korzystamy z metody `registerReceiver()`, przekazując jej obiekt `BroadcastReceiver` oraz obiekt `IntentFilter`:

```
registerReceiver(myReceiver, intentFilter);
```

Mamy teraz zarejestrowany obiekt `BroadcastReceiver`, więc możemy wysłać komunikat, aby sprawdzić, czy wszystko działa. Aby wysłać komunikat, korzystamy z metody `sendBroadcast()`, przekazując jej obiekt `Intent`:

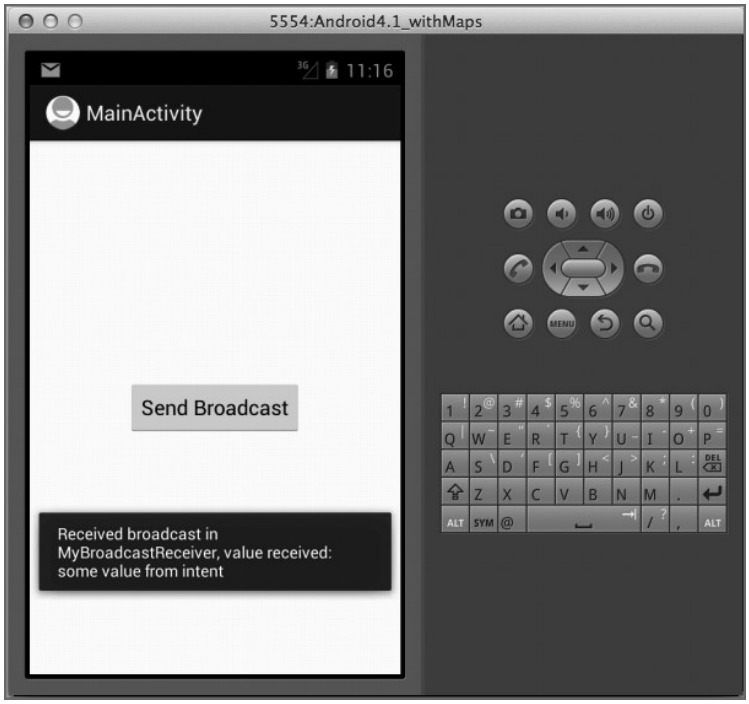
```
public void onClick(View view) {
    Intent i = new Intent("MY_SPECIFIC_ACTION");
    i.putExtra("key", "some value from intent");
    sendBroadcast(i);
}
```

Jeśli chcemy przekazać dane odbiornikowi, możemy użyć metody `putExra()`. Aby wyrejestrować odbiornik komunikatów, używamy metody `unregisterReceiver()`:

```
unregisterReceiver(myReceiver);
```

Na rysunku 1-4 pokazano odbiornik otrzymujący komunikat.

Odbiornik komunikatów będzie działał nawet wtedy, gdy komunikat został wysłany przez inną aplikację.



Rysunek 1-4

Rejestrowanie odbiornika komunikatów w pliku AndroidManifest.xml

W poprzednim przykładzie odbiornik komunikatów nie będzie już działał, jeśli aplikacja pracuje w tle, gdyż po jej przejściu do pracy w tle zostaje przez nas wyrejestrowany:

```
@Override
public void onPause() {
    super.onPause();
    //--- Wyrejestruj odbiornik ---
    unregisterReceiver(myReceiver);
}
```

Jeśli chcemy uzyskać bardziej trwały sposób otrzymywania komunikatów, musimy zarejestrować klasę `BroadcastReceiver` w pliku `AndroidManifest.xml`.

Aby tego dokonać, tworzymy klasę `BroadcastReceiver` w innej klasie Javy. Poniższy fragment kodu pokazuje zawartość pliku `MySecondBroadcastReceiver.java`:

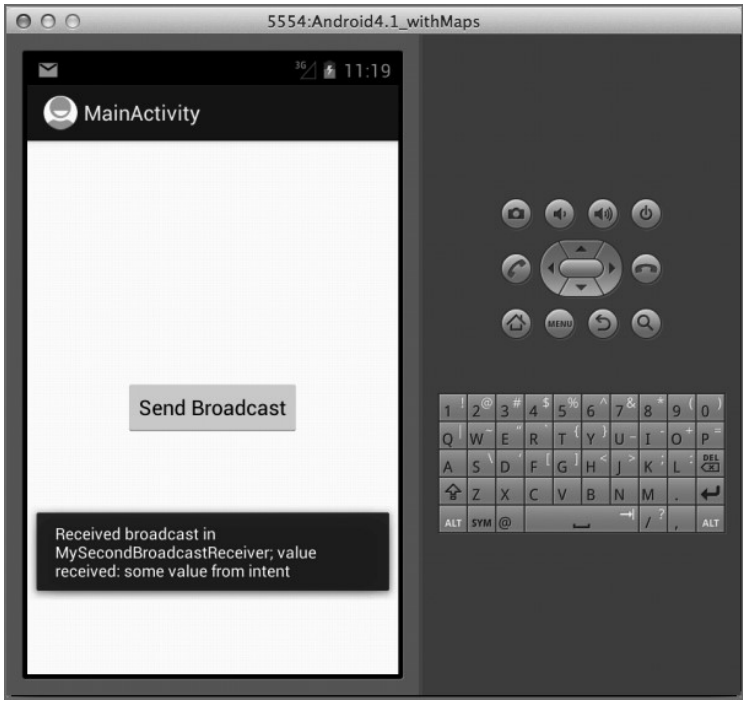
```
package net.learn2develop.usingbroadcastreceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MySecondBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent i) {
        Toast.makeText(context,
```

```
                "Received broadcast in MySecondBroadcastReceiver; " +
                " value received: " + i.getStringExtra("key"),
                Toast.LENGTH_LONG).show();
    }
}
```

Aby zarejestrować ten odbiornik w pliku `AndroidManifest.xml`, dodajemy element `<receiver>`:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.usingbroadcastreceiver"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver android:name=".MySecondBroadcastReceiver" >
            <intent-filter>
                <action android:name="MY_SPECIFIC_ACTION" />
            </intent-filter>
        </receiver>

    </application>

</manifest>
```

Nasza aplikacja ma teraz dwa obiekty `BroadcastReceiver`: jeden zarejestrowany programowo w metodzie `onResume()` oraz drugi w pliku `AndroidManifest.xml`. Jeśli teraz wyślemy komunikat, zostaną wywołane oba odbiorniki. Rysunek 1-5 przedstawia wywołanie drugiego odbiornika.



Rysunek 1-5

Przepis 1.5 Przypisywanie priorytetów odbiornikom komunikatów

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

Brak

Kod źródłowy do pobrania z witryny Wrox.com

UsingBroadcastReceiver.zip

Gdy wysyłamy komunikat za pomocą metody `sendBroadcast()`, wszystkie odbiorniki komunikatów, które pasują do podanego działania, są wywoływane w sposób losowy. Co jednak, jeśli chcemy przypisać określoną kolejność odbiornikom komunikatów, tak aby niektóre z nich były wywoływane przed innymi? Aby to zrobić, trzeba przypisać priorytety odbiornikom komunikatów.

Rozwiązanie

Aby programowo przypisać priorytet odbiornikowi komunikatów, korzystamy z metody `setPriority()`:

```
public class MainActivity extends Activity {
    MyBroadcastReceiver myReceiver;
    IntentFilter intentFilter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        myReceiver = new MyBroadcastReceiver();
        intentFilter = new IntentFilter("MY_SPECIFIC_ACTION");
    }

    @Override
    public void onResume() {
        super.onResume();
        intentFilter.setPriority(10);
        registerReceiver(myReceiver, intentFilter);
    }
```

W metodzie `setPriority()` używane są priorytety od 0 (domyślny) do 1000. Im większa liczba, tym wyższy priorytet, a więc odbiorniki komunikatów z wyższym priorytetem będą wywoływanie przed tymi o priorytecie niższym. Jeśli więcej niż jeden odbiornik komunikatów ma ten sam priorytet, są one wywoływane losowo. Poprzedni fragment kodu ustawia priorytet na 10.

Aby ustawić priorytet odbiorników komunikatów w pliku `AndroidManifest.xml`, używamy atrybutu `android:priority`:

```
<receiver android:name=".MySecondBroadcastReceiver" >
    <intent-filter android:priority="50">
        <action android:name="MY_SPECIFIC_ACTION" />
    </intent-filter>
</receiver>
```

W powyższym przykładzie priorytet jest ustawiony na 50.

Aby wysłać komunikat dostarczany najpierw do odbiorników komunikatów o wyższym priorytecie, nie możemy korzystać z metody `sendBroadcast()`. Zamiast niej musimy użyć metody `sendOrderedBroadcast()`, przekazując jej obiekt `Intent` wraz z wszelkimi dodatkowymi uprawnieniami, które musi mieć odbiornik, aby otrzymać komunikat:

```
public void onClick(View view) {
    Intent i = new Intent("MY_SPECIFIC_ACTION");
    i.putExtra("key", "some value from intent");
```

```
        //sendBroadcast(i);

        //--- Pozwala przerwać komunikat ---
        //--- Pozwala odbiornikom komunikatów na ustawianie priorytetów ---
        sendOrderedBroadcast(i, null);
    }
```

Jeśli spróbujemy teraz wysłać komunikat, zauważymy, że odbiornik podany w pliku `AndroidManifest.xml` jest wywoływany jako pierwszy, przed tym zadeklarowanym programowo w kodzie.

Jeśli chcemy wysłać komunikat tylko do odbiorników komunikatów z uprawnieniem dostępu do Internetu, podajemy uprawnienie jako drugi argument metody `sendOrderedBroadcast()`, w następujący sposób:

```
sendOrderedBroadcast(i, "android.permission.INTERNET");
```

Przerwanie komunikatu

Gdy komunikaty są wysyłane przy użyciu metody `sendOrderedBroadcast()`, odbiorniki komunikatów są wywoływane w kolejności zdefiniowanych priorytetów. Gdy odbiornik komunikatu o wyższym priorytecie otrzyma komunikat, obsłuży go i przekaże do następnego odbiornika komunikatów w kolejce. W niektórych scenariuszach możemy obsłużyć komunikat i nie przekazywać go do kolejnego odbiornika. Aby tego dokonać, możemy użyć metody `abortBroadcast()`:

```
package net.learn2develop.UsingBroadcastReceiver;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class MySecondBroadcastReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent i) {
        Toast.makeText(context,
                "Received broadcast in MySecondBroadcastReceiver; " +
                "value received: " + i.getStringExtra("key"),
                 Toast.LENGTH_LONG).show();
        //--- Przerwij komunikat ----
        abortBroadcast();
    }
}
```

W powyższym fragmencie kodu, klasa `MySecondBroadcastReceiver` przerwie komunikat po otrzymaniu go. Gdy komunikat zostanie przerwany, inne odbiorniki czekające na swoją kolej nie będą go mogły otrzymać.

> **UWAGA** *Aby wywołać metodę* `abortBroadcast()` *w celu przerwania komunikatu, musimy wysłać komunikat za pomocą metody* `sendOrderedBroadcast()`. *Użycie metody* `sendBroadcast()` *nie ma wpływu na priorytet i nie spowoduje przerwania obsługi komunikatu.*

Przepis 1.6 Automatyczne uruchamianie aplikacji w czasie rozruchu

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

android.permission.RECEIVE_BOOT_COMPLETED

Kod źródłowy do pobrania z witryny Wrox.com

AutoStartApp.zip

Jeśli chcemy automatycznie uruchamiać swoje aplikacje przy każdym uruchomieniu urządzenia, musimy zarejestrować `BroadcastReceiver`. W tym przepisie pokazano, jak to zrobić.

Rozwiązanie

Aby automatycznie uruchomić naszą aplikację podczas rozruchu urządzenia, dodajemy nową klasę do naszego pakietu i zapewniamy rozszerzenie klasy podstawowej `BroadcastReceiver`. Przykładem jest poniższa klasa `BootupReceiver`:

```
package net.learn2develop.autostartapp;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class BootupReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, "App started", Toast.LENGTH_LONG).show();
        //--- Uruchom podstawowe działanie naszej aplikacji ---
```

```
        Intent i = new Intent(context,MainActivity.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
    }
}
```

Podczas rozruchu urządzenia spowoduje to uruchomienie odbiornika komunikatów i wywołanie metody `onReceiver()`. Aby wyświetlić nasze działanie podczas rozruchu urządzenia, skorzystamy z obiektu `Intent`. Pamiętajmy, aby do obiektu `Intent` dodać flagę `FLAG_ACTIVITY_NEW_TASK`.

Aby zarejestrować odbiornik komunikatów, musimy dodać element `<receiver>` do pliku `AndroidManifest.xml`. Trzeba także mieć uprawnienie `RECEIVE_BOOT_COMPLETED`:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.autostartapp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />

    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/title_activity_main" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver android:name=".BootupReceiver">
           <intent-filter>
              <action android:name="android.intent.action.BOOT_COMPLETED" />
              <category android:name="android.intent.category.DEFAULT" />
           </intent-filter>
        </receiver>

    </application>

</manifest>
```

Teraz nasza aplikacja zostanie automatycznie uruchomiona podczas rozruchu urządzenia.

Przepis 1.7 Wywoływanie wbudowanych aplikacji

Wersja systemu Android

Poziom 1 i wyższe

Uprawnienia

Brak

Kod źródłowy do pobrania z witryny Wrox.com

CallingApps.zip

Jedną z kluczowych funkcji systemu Android jest zapewnienie bezproblemowego wywoływania z aplikacji innych aplikacji. Pozwala to na zintegrowanie na urządzeniu różnych aplikacji, tworząc spójny system dla użytkowników. Ten przepis pokazuje różne sposoby wywoływania aplikacji na naszym urządzeniu.

Rozwiązanie

Jest wiele sposobów wywoływania wbudowanych aplikacji, zaś wybór sposobu zależy od tego, którą aplikację wywołujemy. W rozwiązaniu tego przepisu nauczymy się, jak wywoływać niektóre często instalowane aplikacje na naszym urządzeniu z systemem Android, takie jak:

- wyświetlanie map
- przekierowanie użytkownika do określonej aplikacji w Google Play
- wysyłanie widomości e-mail
- wysyłanie zawartości tekstowych i graficznych do aplikacji, które je obsługują

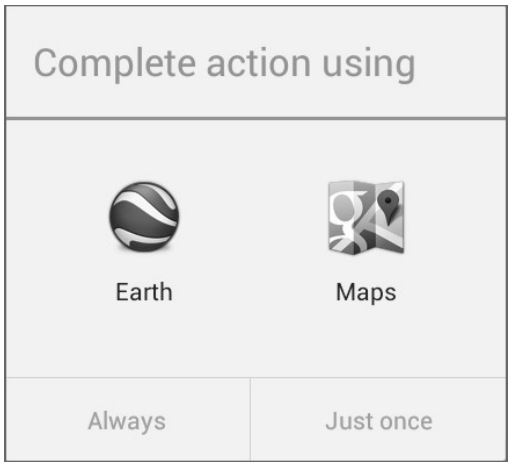
Wyświetlanie map

Aby w naszej aplikacji wyświetlić mapy, możemy uruchomić działanie korzystające ze schematu `geo:scheme`, jak pokazano poniżej:

```
package net.learn2develop.callingapps;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
```

```
public class MainActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Intent i = new Intent(android.content.Intent.ACTION_VIEW);
        i.setData(Uri.parse("geo:37.827500,-122.481670"));
        startActivity(i);
    }

}
```

Na rysunku 1-6 pokazano aplikację wyświetlającą listę aplikacji obsługujących schemat `geo:scheme`, gdy powyższy kod działa na urządzeniu z systemem Android (na swoim urządzeniu można zobaczyć więcej informacji).



Rysunek 1-6

Aby uruchomić aplikację z tej listy, wybieramy ją (na przykład Google Earth) i wybieramy opcję Always (Zawsze) lub Just once (Tylko raz). Jeśli wybierzemy Always, wybrana aplikacja (w tym przypadku Google Earth) będzie zawsze uruchamiana automatycznie. Jeśli wybierzemy Just once, zobaczymy pokazane zgłoszenie (z pytaniem, czy chcemy uruchomić Always or Just once) przy każdym uruchomieniu tej aplikacji.

Gdy wybierzemy aplikację Earth, zostanie uruchomiona aplikacja Google Earth (patrz rysunek 1-7). Podobnie, wybranie aplikacji Maps spowoduje uruchomienie aplikacji Google Maps (patrz rysunek 1-8).