# Android

## Podstawy tworzenia aplikacji

Piotr Fulmański

Instytut Nauk Ekonomicznych i Informatyki,
Państwowa Wyższa Szkoła Zawodowa w Płocku, Polska

March 4, 2015

# Table of contents

# Framework

Jednym z najwartościowszych aspektów platform takich jak Android jest **framework aplikacji**.
Przede wszystkim zapewnia on istnienie pewnej wspólnej dla wszystkich aplikacji struktury, w którą można je „wpasować".
Korzystanie z frameworków to jak przyjęcie umowy. Jeśli umieścimy nasze pliki w odpowiednich miejscach, framework wykorzysta te pliki w znany jemu, określony sposób. Otrzymujemy w ten sposób pewien schemat, którym możemy się posługiwać. Wówczas wiele szablonowych zadań wykonywanych jest za programistę, który dzięki temu może skoncentrować się na aplikacji a nie na pewnych powtarzalnych detalach (które często wcale takimi detalami nie są).

Typowe elementy struktury projektu

```
app/src/main/res/layout/activity_my.xml
                   /drawable<density>/
                   /layout/
                   /menu/
                   /values/
app/src/main/java/com.mycompany.myfirstapp/MyActivity.java
app/src/main/AndroidManifest.xml
```

# Podstawowe elementy aplikacji

Aplikacje na Android obejmują kilka podstawowych elementów.

- Aktywność (obiekt klasy Activity ). Działa na pierwszym planie. Zarządza interfejsem użytkownika oraz obsługuje zdarzenia i interakcje.
- Usługa (obiekt klasy Service ). Działa w tle. Może obsługiwać długotrwałe zadania lub monitorowanie.
- Odbiornik (obiekt klasy BroadcastReceiver). Obejmuje metody obsługi zdarzeń uruchamiane przez rozpowszechniane zdarzenia (intencje) i reagujące na nie.
- Dostawca treści (obiekt klasy ContentProvider). Umożliwia udostępnianie innym aplikacjom interfejsu API do wymiany danych.

# Podstawowe elementy aplikacji

Aplikacje na Android obejmują kilka podstawowych elementów.

- Aktywność (obiekt klasy Activity ). Działa na pierwszym planie. Zarządza interfejsem użytkownika oraz obsługuje zdarzenia i interakcje.
- Usługa (obiekt klasy Service ). Działa w tle. Może obsługiwać długotrwałe zadania lub monitorowanie.
- Odbiornik (obiekt klasy BroadcastReceiver). Obejmuje metody obsługi zdarzeń uruchamiane przez rozpowszechniane zdarzenia (intencje) i reagujące na nie.
- Dostawca treści (obiekt klasy ContentProvider). Umożliwia udostępnianie innym aplikacjom interfejsu API do wymiany danych.

# Podstawowe elementy aplikacji

Aplikacje na Android obejmują kilka podstawowych elementów.

- Aktywność (obiekt klasy Activity ). Działa na pierwszym planie. Zarządza interfejsem użytkownika oraz obsługuje zdarzenia i interakcje.
- Usługa (obiekt klasy Service ). Działa w tle. Może obsługiwać długotrwałe zadania lub monitorowanie.
- Odbiornik (obiekt klasy BroadcastReceiver). Obejmuje metody obsługi zdarzeń uruchamiane przez rozpowszechniane zdarzenia (intencje) i reagujące na nie.
- Dostawca treści (obiekt klasy ContentProvider). Umożliwia udostępnianie innym aplikacjom interfejsu API do wymiany danych.

# Podstawowe elementy aplikacji

Aplikacje na Android obejmują kilka podstawowych elementów.

- Aktywność (obiekt klasy Activity ). Działa na pierwszym planie. Zarządza interfejsem użytkownika oraz obsługuje zdarzenia i interakcje.
- Usługa (obiekt klasy Service ). Działa w tle. Może obsługiwać długotrwałe zadania lub monitorowanie.
- Odbiornik (obiekt klasy BroadcastReceiver). Obejmuje metody obsługi zdarzeń uruchamiane przez rozpowszechniane zdarzenia (intencje) i reagujące na nie.
- Dostawca treści (obiekt klasy ContentProvider). Umożliwia udostępnianie innym aplikacjom interfejsu API do wymiany danych.

# Komponenty łączące

Do tworzenia podstawowych elementów aplikacji i ich łączenia służą dodatkowe komponenty.

- Widoki (obiekty klasy View). Są to elementy interfejsu użytkownika wyświetlane na ekranie.
- Układy (obiekty klasy Layout). Są to hierarchie widoków kontrolujące format i wygląd ekranu.
- Intencje (obiekty klasy Intent). Są to komunikaty łączące komponenty ze sobą.
- Zasoby (obiekty klasy Resource). Są to elementy zewnętrzne, na przykład łańcuchy znaków i obiekty graficzne (obrazki).
- Manifest (obiekt klasy Manifest). Obejmuje konfigurację aplikacji.

# Komponenty łączące

Do tworzenia podstawowych elementów aplikacji i ich łączenia służą dodatkowe komponenty.

- Widoki (obiekty klasy View). Są to elementy interfejsu użytkownika wyświetlane na ekranie.
- Układy (obiekty klasy Layout). Są to hierarchie widoków kontrolujące format i wygląd ekranu.
- Intencje (obiekty klasy Intent). Są to komunikaty łączące komponenty ze sobą.
- Zasoby (obiekty klasy Resource). Są to elementy zewnętrzne, na przykład łańcuchy znaków i obiekty graficzne (obrazki).
- Manifest (obiekt klasy Manifest). Obejmuje konfigurację aplikacji.

# Komponenty łączące

Do tworzenia podstawowych elementów aplikacji i ich łączenia służą dodatkowe komponenty.

- Widoki (obiekty klasy View). Są to elementy interfejsu użytkownika wyświetlane na ekranie.
- Układy (obiekty klasy Layout). Są to hierarchie widoków kontrolujące format i wygląd ekranu.
- Intencje (obiekty klasy Intent). Są to komunikaty łączące komponenty ze sobą.
- Zasoby (obiekty klasy Resource). Są to elementy zewnętrzne, na przykład łańcuchy znaków i obiekty graficzne (obrazki).
- Manifest (obiekt klasy Manifest). Obejmuje konfigurację aplikacji.

# Komponenty łączące

Do tworzenia podstawowych elementów aplikacji i ich łączenia służą dodatkowe komponenty.

- Widoki (obiekty klasy View). Są to elementy interfejsu użytkownika wyświetlane na ekranie.
- Układy (obiekty klasy Layout). Są to hierarchie widoków kontrolujące format i wygląd ekranu.
- Intencje (obiekty klasy Intent). Są to komunikaty łączące komponenty ze sobą.
- Zasoby (obiekty klasy Resource). Są to elementy zewnętrzne, na przykład łańcuchy znaków i obiekty graficzne (obrazki).
- Manifest (obiekt klasy Manifest). Obejmuje konfigurację aplikacji.

# Komponenty łączące

Do tworzenia podstawowych elementów aplikacji i ich łączenia służą dodatkowe komponenty.

- Widoki (obiekty klasy View). Są to elementy interfejsu użytkownika wyświetlane na ekranie.
- Układy (obiekty klasy Layout). Są to hierarchie widoków kontrolujące format i wygląd ekranu.
- Intencje (obiekty klasy Intent). Są to komunikaty łączące komponenty ze sobą.
- Zasoby (obiekty klasy Resource). Są to elementy zewnętrzne, na przykład łańcuchy znaków i obiekty graficzne (obrazki).
- Manifest (obiekt klasy Manifest). Obejmuje konfigurację aplikacji.

# Manifest

Manifest to plik, w którym zdefiniowane są relacje, możliwości, uprawnienia i konfiguracja każdej aplikacji na Android.
Manifest aplikacji jest punktem wyjścia do tworzenia (uruchamiania) każdej aplikacji na Android. Pierwszą rzeczą, jaką wykonuje system operacyjny Androida podczas uruchamiania aplikacji jest wczytanie manifestu aplikacji ponieważ to manifest określa, które fragmenty kodu należy uruchomić. W świecie Androida jednostkami pracy są aktywności. W pliku manifestu aplikacji trzeba wskazać, która klasa aktywności jest punktem wejścia do aplikacji.

# Manifest

Listing 2.1. Plik manifestu AndroidManifest.xml aplikacji DealDroid

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="com.manning.aip.dealdroid"
android:versionCode="1"
android:versionName="1.0">
<application
android:icon="@drawable/ddicon"
android:label="@string/app_name"
android:name=".DealDroidApp">
<activity
android:name=".DealList"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity
android:name=".DealDetails"
android:label="@string/deal_details" />
</application>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-sdk android:minSdkVersion="4" />
</manifest>
```

# Uprawnienia

System uprawnień Androida opisuje wszystkie zadeklarowane w ten sposób zastrzeżone operacje, a następnie wyświetla je użytkownikowi, gdy jest instalowana aplikacja.
Można zadeklarować takie operacje, jak odczyt lub zapis danych w systemie plików, odczyt lub zapis danych kontaktowych użytkownika, możliwość wzbudzenia telefonu, dostęp do sieci (internetu) itd.
Znacznie rzadziej potrzebne jest deklarowanie i wymuszanie własnych niestandardowych uprawnień, wykraczających poza deklaracje systemowe.
Jeśli zachodzi potrzeba wykorzystania takich uprawnień, można je zadeklarować w manifeście i wymuszać je w komponentach (aktywnościach, usługach, odbiornikach rozpowszechnianych informacji itd.).

# Zasoby

Zasób to bardzo ogólne określenie. Jako zasoby można definiować m.in menu, style, animacje, kształty, tablice danych i inne elementy. Zasoby są definiowane przez umieszczanie plików w katalogu `/res` projektu. Następnie dostęp do zasobów można uzyskać albo w kodzie, albo przez wskazywanie ich w plikach XML.

# Zasoby - cechy

Wszystkie elementy zadeklarowane w katalogu `/res` nie tylko są pakowane wraz z aplikacją, ale są też programowo dostępne w kodzie aplikacji. Zasoby mają kilka kluczowych cech, o których warto pamiętać:

- każdy zasób ma identyfikator;
- każdy zasób ma określony typ;
- zasoby mają określoną lokalizację i plik, w którym są zdefiniowane.

# Zasoby – cechy

- Zasoby pozwalają oddzielić kod od zewnętrznych elementów, takich jak obrazki i łańcuchy znaków. Taki podział jest pożyteczny, ponieważ kod jest wtedy konkretny i przejrzysty.
- Zasoby są wydajne i szybkie. Zasoby w formacie XML są kompilowane na format binarny. Dzięki temu są wygodne na etapie programowania i szybkie w czasie wykonywania programu.
- Zasoby umożliwiają obsługę dynamicznego ładowania w trakcie wykonywania programu na podstawie różnych właściwości środowiskowych, takich jak język, konfiguracja ekranu si możliwości sprzętowe.

# Zasoby – przykład: napisy

Przykładowy plik zasobów `res/values/strings.xml` z wartościami nazwanych łańcuchów znaków

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

Wszystkie łańcuchy znaków zdefiniowane w pliku `strings.xml` są zapisane jako stałe w klasie `R`

# Zasoby – przykład: układ ekranu

Przykładowy plik zasobów `res/layout/activity_my.xml` opisujący układ ekranu

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
      <EditText android:id="@+id/edit_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
      <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send" />
</LinearLayout>
```

# Dostęp do zasobów

Zdefiniowane zasoby można wskazywać w kodzie lub w XML-u za pomocą odpowiednich identyfikatorów.
Aby uzyskać dostęp do zasobu w kodzie, należy użyć zdefiniowanego w klasie `R` identyfikatora, na przykład `R.string.deal_details` do zasobu zdefiniowanego przez nas lub `android.R.string.yes` do zasobu systemowego. Zwykle takie identyfikatory stosuje się w metodach klasy `Resources`: `Resources.getString(R.string.deal_details)`.
Aby uzyskać dostęp do zasobu w XML-u wystarczy poprzedzić identyfikator potrzebnego zasobu przedrostkiem `@`. Łańcuch znaków `deal_details` można wskazać tak: `@string/deal_details` a zasób systmowy tak: `@android:string/yes`.

# Układ, widoki i kontrolki

Układ to specjalny zasób, który służy do projektowania ekranów, tworzenia pozycji na listach i innych elementów interfejsu użytkownika w Androidzie.

# Aktywności

Aktywność (klasa typu `Activity`) odpowiada jednej konkretnej rzeczy, jaką użytkownik może zrobić. Zwykle każdy ekran aplikacji jest definiowany za pomocą układu i składa się z widoków oraz kontrolek obsługiwanych przez powiązaną aktywność. Każda aktywność

- tworzy okno interfejsu użytkownika,
- zarządza cyklem życia i stanem,
- stanowi punkt docelowy intencji,
- obsługuje zdarzenia interfejsu,
- kontroluje menu itd.

# Aktywności – podstawowe informacje

Oto najczęściej używane metod aktywności:

- Metoda onCreate. Wywoływana przy tworzeniu obiektu klasy Activity .
- Metoda onPause. Wywoływana przy odsuwaniu obiektu klasy Activity w tło.
- Metoda onResume. Wywoływana przy wznawianiu pracy obiektu klasy Activity działającego w tle.

# Adaptery

Przekazywanie danych ze źródła do widoku odbywa się za pośrednictwem obiektu klasy Adapter. Jak wskazuje nazwa, klasa ta adaptuje określone źródło danych, a tym samym pozwala podłączać różne rodzaje źródeł danych do widoku (obiektu klasy AdapterView), który następnie wyświetla dane na ekranie.

# Intencje i filtry intencji

Jedną z wyjątkowych zalet Androida jest elastyczność w zakresie komunikacji między komponentami i wymiany danych między nimi. Android umożliwia to na podstawie zdarzeń opartych na klasach `Intent` i `IntentFilter`.
Platforma Androida śledzi wszystkie deklaracje IntentFilter dostępne w działającym systemie, a następnie dynamicznie (w czasie wykonywania programu) łączy zgłaszane intencje z najodpowiedniejszymi komponentami.

# Podsumowanie

Podstawowymi cegiełkami aplikacji na Android są manifest aplikacji, zasoby, układy, widoki, aktywności i intencje.
Manifest obejmuje konfigurację aplikacji, a zasoby to przechowywane zewnętrznie elementy (na przykład łańcuchy znaków i rysunki). Kod pojawia się w aktywnościach, które pobierają zasoby i układy. Układy to zbiory widoków porządkujące interfejs użytkownika na ekranie lub w komponencie. Układy często są zapisywane w formacie XML i przekształcane na klasy w czasie wykonywania programu, co dodatkowo pomaga rozdzielać elementy aplikacji. Aktywności na podstawie widoków i kontrolek tworzą wyświetlane użytkownikowi elementy, z którymi można wchodzić w interakcje. Intencje łączą komponenty, a nawet różne aplikacje.