ЛЕКЦІЯ 5. МАНІФЕСТ ТА ПРИЙМАЧІ ШИРОКОМОВНИХ ПОВІДОМЛЕНЬ (BROADCAST RECEIVERS)

Кожен широкомовний приймач є спадкоємцем класу BroadcastReceiver. Цей клас розрахований на отримання об'єктів-намірів відправлених методом sendBroadcast().

Можна виділити два різновиди широкомовних повідомлень:

- **Нормальні широкомовні повідомлення** передаються за допомогою Context.sendBroadcast в асинхронному режимі. Всі приймачі спрацьовують у невизначеному порядку, часто в один і той же час.
- **Спрямовані широкомовні повідомлення** передаються за допомогою Context.sendOrderedBroadcast тільки одному приймачу в один момент часу. Як тільки приймач спрацює, він може передати повідомлення наступного приймачу, а може перервати мовлення так, що більше жоден приймач це повідомлення не отримає.

Навіть у випадку нормального широкомовлення можуть скластися ситуації, в яких система буде передавати повідомлення одному приймачу в один момент часу. Особливо це актуально для приймачів, які вимагають створення процесів, щоб не перевантажувати систему новими процесами. Однак у цьому випадку жоден приймач не може перервати широкомовлення.

Об'єкт типу BroadcastReceiver дійсний тільки під час виклику методу onRecieve() , як тільки метод виконаний, система завершує роботу об'єкта і більше не активує його.

Детальніше про приймачах широкомовних повідомлень:

http://developer.android.com/reference/android/content/BroadcastReceiver.html

МАНІФЕСТ

Кореневий каталог кожної програми під Android повинен містити файл AndroidManifest.xml(в точності з такою назвою). Маніфест додатки містить всю необхідну інформацію, використовувану системою для запуску і виконання програми. Основна інформація , що міститься в маніфесті:

- Ім'я Java пакету додатку, який використовується як унікальний ідентифікатор додатки.
- Опис компонентів програми: активностей, сервісів, приймачів широкомовних повідомлень і контент-провайдерів, які складають додаток. Для кожного компонента додатка визначено ім'я відповідного класу і оголошені їхні основні властивості(наприклад, з якими повідомленнями-намірами вони можуть працювати). Ця інформація дозволяє системі Android дізнатися які компоненти і за яких умов можуть бути запущені.
- Визначення процесів, в яких будуть виконуватися компоненти програми.
- Оголошення повноважень, якими має володіти додаток для доступу до захищених частинах API та взаємодії з іншими додатками.
- Оголошення повноважень, якими повинні володіти інші програми для взаємодії з компонентами даного.
- Список допоміжних класів, які надають інформацію про хід виконання програми. Ці оголошення містяться в маніфесті поки йде розробка та налагодження програми, перед публікацією додатки вони віддаляються.
- Визначення мінімального рівня Android API для програми.
- Список бібліотек пов'язаних з додатком.

У файлі маніфесту тільки два елементи: <manifest> і <application> є обов'язковими і при цьому зустрічаються рівно по одному разу. Інші елементи можуть зустрічатися декілька разів або не відображатись зовсім, в цьому випадку маніфест визначає пусте додаток.

Наступний лістинг демонструє загальну структуру файлу маніфесту.

```
 <? Xml version = "1.0" encoding = "utf-8"?>
<Manifest>
 <Uses-permission />
 <Permission />
 <Permission-tree />
 <Permission-group />
 <Instrumentation />
 <Uses-sdk />
 <Uses-configuration />
 <Uses-feature />
 <Support-screens />
 <Compatible-screens />
 <Supports-gl-texture />
```

```
<Application>
 <Activity>
  <Intent-filter>
   <Action />
   <Category />
   <Data />
  </ Intent-filter>
  <Meta-data />
 </ Activity>
 <Activity-alias>
  <Intent-filter>... </ intent-filter>
  <Meta-data />
 </ Activity-alias>
 <Service>
  <Intent-filter>... </ intent-filter>
  <Meta-data />
 </ Service>
 <Receiver>
  <Intent-filter>... </ intent-filter>
  <Meta-data />
 </ Receiver>
 <Provider>
  <Grant-uri-permission />
  <Meta-data />
  <Path-permission />
 </ Provider>
 <Uses-library />
</ Application>
</ Manifest>
```

Лістинг 1. Структура файлу AndroidManifest.xml

У маніфесті елементи одного рівня, такі як <activity> , <service> , <receiver> , <provider> , можуть слідувати один за одним в будь-якій послідовності. Елемент <activity-alias> є виключенням з цього правила, він повинен слідувати за відповідною активністю.

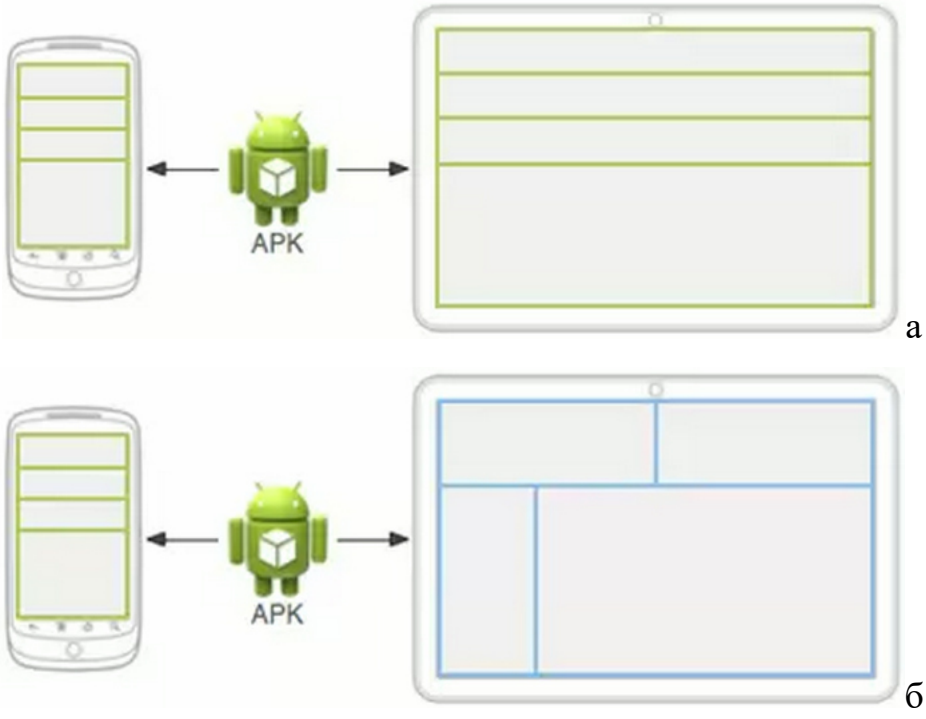
РЕСУРСИ

При розробці мобільних додатків необхідно виробити звичку відокремлювати ресурси додатки від коду. До ресурсів додатки можуть належати: зображення, рядки, кольору, компоновки елементів інтерфейсу користувача(layout) і т.д. Відділення ресурсів від коду дозволяє використовувати альтернативні ресурси для різних конфігурацій пристроїв: мова, дозвіл екрана і т.д.

Для забезпечення сумісності з різними конфігураціями, ресурси необхідно згрупувати в директорії за типом ресурсів та конфігурації пристрою, отримані директорії помістити в папку **res/**.

Для будь-якого типу ресурсів можна визначити дві групи. Перша визначає ресурси, які будуть використовуватися незалежно від конфігурації пристрою або в тому випадку, коли під конфігурацію немає відповідних альтернативних ресурсів. Ця група називається ресурси за замовчуванням(default).

Друга група визначає ресурси, які підходять для певної конфігурації пристрою, розміщується в директорії з назвою, що позначає дану конфігурацію. Такі ресурси називаються альтернативними.



а

б

Рис. 1. Використання ресурсів: а) використовується компоновка за замовчуванням(додаток не містить альтернативи); б) кожен пристрій використовує відповідну компоновку

Кожен тип ресурсів необхідно розміщувати в спеціальній піддиректорії папки **res /**. Розглянемо основні з цих піддиректорій:

animator /	- Містить XML файли, які визначають властивості анімації;
anim /	- Містить XML файли, які визначають анімацію перетворень;
color /	- Містить XML файли, які визначають списки кольорів;
drawable/	- Містить графічні файли або XML файли, які компілюються в графічні ресурси;
layout /	- Містить XML файли, які визначають компоновку елементів користувальницького інтерфейсу;
menu /	- Містить XML файли, які визначають все меню додатка;
values /	- Містить XML файли, які визначають прості значення, таких ресурсів як, рядки, числа, кольору.

Слід зазначити, що файли ресурсів не можна розміщувати в папку **res /** безпосередньо, вони обов'язково повинні розміщуватися у відповідному каталозі, інакше буде видана помилка компіляції.

Всі ресурси, які містяться в розглянутих піддиректоріях є ресурсами за замовчуванням. Зрозуміло, що різні типи пристроїв можуть вимагати різних типів ресурсів.

Наприклад, для пристроїв з різними розмірами екрану компоновки елементів користувальницького інтерфейсу повинні відрізнятися. Рис 1 показує варіанти зовнішнього вигляду програми з використанням тільки компоновки за замовчуванням(а) і з використанням альтернативних компоновок(б). Навіть на схемі зрозуміло, що при правильному підході додаток , що змінює свій зовнішній вигляд залежно від розміру екрана привабливіше, ніж залишається незмінним.

Щоб визначити залежні від конфігурації альтернативи для безлічі ресурсів:

1. необхідно створити директорію в каталозі **res /** , присвоїти цій директорії ім'я в такій формі: ім'я_ресурсу-специфікатор_конфігурації, де

- ім'я_ресурсу - ім'я директорії, відповідного ресурсу за замовчуванням(див. вище);

- специфікатор_конфігурації - ім'я, що визначає конфігурацію, для якої використовуються дані ресурси. Повний список доступних

2. необхідно зберегти ресурси в новій директорії, файл ресурсів повинен називатися в точності так само, як відповідний файл ресурсів за замовчуванням.

Наприклад, якщо компоновка елементів користувальницького інтерфейсу збережена, як ресурс за замовчуванням, в папці **res / layout /** , можна(скоріше навіть потрібно) визначити альтернативну компоновку елементів користувальницького інтерфейсу, відповідну горизонтальній(альбомній) орієнтації екрану смартфона і зберегти її в папці **res / layout-land /**. Android автоматично визначить відповідну компоновку, звіряючи поточний стан пристрою з іменами папок в каталозі **/ res**.

Всі ресурси після визначення можуть бути доступні за посиланням на їх ID , які визначені в автоматично генерованому класі R. Для кожного типу ресурсів в R класі існує підклас , наприклад, R.drawable для всіх графічних ресурсів. ID ресурсу завжди має дві складові:

- тип ресурсу - всі ресурси групуються за типами, наприклад, string, drawable, layout;
- ім'я ресурсу - або ім'я файлу без розширення, яке значення атрибуту android: name в XML файлі для простого значення.

Отримати доступ до ресурсу можна двома способами:

- в коді: можна використовувати вирази виду R.тіп_ресурса.імя_ресурса, наприклад, R.string.hello ;
- в XML: використовується спеціальний XML синтаксис, який відповідає ID визначеному в R класі, наприклад, @ string / hello.