ЛЕКЦІЯ 3. ВИДИ ДОДАТКІВ І ЇХ СТРУКТУРА

2.1 ВВЕДЕННЯ

У даній лекції розглядається архітектура Android додатків, заснована на ідеї багаторазового використання компонентів, які є основними будівельними блоками. Докладно описуються основні компоненти, а також такі важливі поняття для мобільних додатків, що працюють під управлінням Android, як маніфест додатки і ресурси.

2.2 ОСНОВНІ ВИДИ ANDROID-ДОДАТКІВ

Приступаючи до розробки мобільних додатків добре б мати уявлення про те, які види додатків існують. Справа в тому, що якщо вдасться визначити до якого типу відноситься додаток, то стає зрозуміліше на які моменти в процесі його розробки необхідно звертати основну увагу. Можна виділити наступні види додатків:

- **Додатки переднього плану** виконують свої функції тільки, коли видимі на екрані, в іншому ж випадку їх виконання призупиняється. Такими додатками є, наприклад, ігри, текстові редактори, відеопрогравачі. При розробці таких програм необхідно дуже уважно вивчити життєвий цикл активності, щоб перемикання в фоновий режим і назад проходили гладко(безшовно), тобто При поверненні додатки на передній план було непомітно, що воно взагалі кудись пропадало. Для досягнення цієї гладкості необхідно стежити за тим, щоб при вході в фоновий режим додаток зберігало свій стан, а при виході на передній план відновлювало його. Ще один важливий момент, на який обов'язково треба звернути увагу при розробці додатків переднього плану, зручний і інтуїтивно зрозумілий інтерфейс.
- **Фонові програми** після настройки не припускають взаємодії з користувачем, більшу частину часу перебувають і працюють в прихованому стані. Прикладами таких додатків можуть служити, служби екранування дзвінків, SMS-автовідповідачі. У більшості своїй фонові програми націлені на відстеження подій, породжуваних апаратним забезпеченням, системою або іншими додатками, працюють непомітно. Можна створювати абсолютно невидимі сервіси, але тоді вони

будуть некерованими. Мінімум дій, які необхідно дозволити користувачеві: санкціонування запуску сервісу, настройка, призупинення і переривання його роботи при необхідності.

- **Змішані додатки** більшу частину часу працюють у фоновому режимі, проте допускають взаємодію з користувачем після. Зазвичай взаємодія з користувачем зводиться до повідомлення про які-небудь події. Прикладами таких додатків можуть служити мультимедіа-програвачі, програми для обміну текстовими повідомленнями(чати), поштові клієнти. Можливість реагувати на користувача введення і при цьому не втрачати працездатності у фоновому режимі є характерною особливістю змішаних додатків. Такі програми зазвичай містять як видимі активності, так і приховані(фонові) сервіси, і при взаємодії з користувачем повинні враховувати свій поточний стан. Можливо буде потрібно оновлювати графічний інтерфейс, якщо додаток знаходиться на передньому плані, або ж посилати користувачеві повідомлення з фонового режиму, щоб тримати його в курсі, що відбувається. І ці особливості необхідно враховувати при розробці подібних додатків.
- **Віджети** - невеликі додатки, які відображаються у вигляді графічного об'єкта на робочому столі. Прикладами можуть служити, додатки для відображення динамічної інформації, такої як заряд батареї, прогноз погоди, дата і час. Зрозуміло, складні додатки можуть містити елементи кожного з розглянутих видів. Плануючи розробку програми, необхідно визначити спосіб його використання, тільки після цього приступати до проектування і безпосередньо розробці.

2.3 БЕЗПЕКА

Звернемо увагу на організацію виконання додатків в ОС Android. Як вже було зазначено додатки під Android розробляються на мові програмування Java, компілюється в файл з розширенням.apk, після цього файл використовується для установки програми на пристрої, що працюють під управлінням Android. Після установки кожне Android додаток "живе" у своїй власній безпечною "пісочниці", розглянемо, як це виглядає:

- операційна система Android є багатокористувацької ОС, в якій кожен додаток розглядається як окремий користувач;
- за замовчуванням, система призначає кожному з додатком унікальний користувальницький ID, який використовується тільки системою і невідомий додатком;
- система встановлює права доступу до всіх файлів додатка наступним чином: доступ до елементів додатки має тільки користувач з відповідним ID;
- кожному додатку відповідає окремий Linux процес, який запускається, як тільки це необхідно хоча б одному компоненту програми, процес припиняє роботу, коли ні один компонент програми не використовує його або ж системі потрібно звільнити пам'ять для інших(можливо, більш важливих) додатків;
- кожному процесу відповідає окремий екземпляр віртуальної машини Dalvik, у зв'язку з цим код додатку виповнюється ізольовано від інших додатків.

Перераховані ідеї функціонування програми в ОС Android реалізують принцип мінімальних привілеїв, тобто кожному додатку, за замовчуванням, дозволений доступ тільки до компонентів, необхідним для його роботи і ніяким більше. Таким чином забезпечується дуже безпечне середовище функціонування додатків.

Однак, у випадку необхідності додатки можуть отримати доступ до даних інших додатків і системним сервісів(послуг). У випадку, коли двом додаткам необхідно мати доступ до файлів один одного, їм присвоюється один і той же користувальницький ID. Для економії системних ресурсів такі додатки запускаються в одному Linux процесі і ділять між собою один і той же екземпляр віртуальної машини, в цьому випадку додатки також повинні бути підписані одним сертифікатом. У разі ж, коли додатку потрібно доступ до системних даними, наприклад, контактам, SMS повідомленнями, картам пам'яті, камері, Bluetooth, Користувачеві необхідно дати додатком такі повноваження під час встановлення його на пристрій.

2.4 АРХІТЕКТУРА ПРОГРАМИ, ОСНОВНІ КОМПОНЕНТИ

Ось і прийшла пора поговорити безпосередньо про внутрішню організацію додатків під Android: обговорити їх архітектуру і основні компоненти.

Архітектура Android додатків заснована на ідеї багаторазового використання компонентів, які є основними будівельними блоками. Кожен компонент є окремою сутністю і допомагає визначити загальний поведінку програми.

Система Android вибудувана таким чином, що будь-яке додаток може запускати необхідний компонент іншої програми. Наприклад, якщо додаток припускає використання камери для створення фотографій, зовсім необов'язково створювати в цьому додатку активність для роботи з камерою. Напевно на пристрої вже є додаток для отримання фотографій з камери, досить запустити відповідну активність, зробити фотографію і повернути її в додаток, так що користувач буде вважати, що камера частина додатку, з яким він працює.

Коли система запускає компонент, вона запускає процес програми, якому належить компонент, якщо він ще не запущений, і створює екземпляри класів, необхідних компоненту. Тому на відміну від більшості інших систем, в системі Android програми не мають єдиної точки входу(немає методу main(), наприклад). В силу запуску кожної програми в окремому процесі і обмежень на доступ до файлів, додаток не може безпосередньо активувати компонент іншої програми. Таким чином для активації компонента іншої програми необхідно надіслати системі повідомлення про намір запустити певний компонент, система активує його.

Можна виділити чотири різних типи компонентів, кожен тип служить для досягнення певної мети і має свій особливий життєвий цикл, який визначає способи створення і руйнування відповідного компонента. Розглянемо основні компоненти Android-додатків.

Активності(Activities). Активність - це видима частина додатки(екран, вікно, форма), відповідає за відображення графічного інтерфейсу користувача. При цьому додаток може мати кілька активностей, наприклад, у додатку, призначеному для роботи з електронною поштою, одна активність може використовуватися для відображення списку нових листів, інша активність - для написання, і ще одна - для читання листів. Незважаючи на те, що для користувача додаток представляється єдиним цілим, все активності додатки не залежать один від одного. У зв'язку з цим

будь-яка з цих активностей може бути запущена з іншої програми, що має доступ до активностям цього додатка. Наприклад, додаток камери може запустити активність, що створює нові листи, щоб відправити щойно зроблену фотографію адресату, зазначеному користувачем.

Сервіси(Services). Сервіс - компонент, який працює у фоновому режимі, виконує тривалі за часом операції або роботу для віддалених процесів. Сервіс не надає інтерфейсу користувача. Наприклад, сервіс може програвати музику у фоновому режимі, поки користувач використовує інше додаток, може завантажувати дані з мережі, що не блокуючи взаємодію користувача з активністю. Сервіс може бути запущений іншим компонентом і після цього працювати самостійно, а може залишитися пов'язаним з цим компонентом і взаємодіяти з ним.

Контент-провайдери(Content providers). Контент-провайдер управляє розподіленим безліччю даних програми. Дані можуть зберігатися в файлової системі, в базі даних SQLite, в мережі, в будь-якому іншому доступному для програми місці. Контент-провайдер дозволяє іншим програмам за наявності у них відповідних прав робити запити або навіть міняти дані. Наприклад, у системі Android є контент-провайдер, який управляє інформацією про контакти користувача. У зв'язку з цим, будь-яке додаток з відповідними правами може зробити запит на читання і запис інформації будь-якого контакту. Контент-провайдер може бути також корисний для читання і запису приватних даних програми, не призначених для доступу ззовні.

Приймачі широкомовних повідомлень(Broadcast Receivers). Приймач - компонент, який реагує на широкомовні повідомлення. Більшість таких оповіщень породжуються системою, наприклад, повідомлення про те, що екран відключився або низький заряд батареї. Додатки також можуть ініціювати широкомовлення, наприклад, розіслати іншим додаткам повідомлення про те, що деякі дані завантажені і доступні для використання. Хоча приймачі не відображають користувача інтерфейсу, вони можуть створювати повідомлення на панелі станів, щоб попередити користувача про появу повідомлення. Такий приймач служить провідником до інших компонентів і призначений для виконання невеликого обсягу робіт, наприклад, він може запустити відповідний події сервіс.

Усі розглянуті компоненти є спадкоємцями класів, визначених у Android SDK.

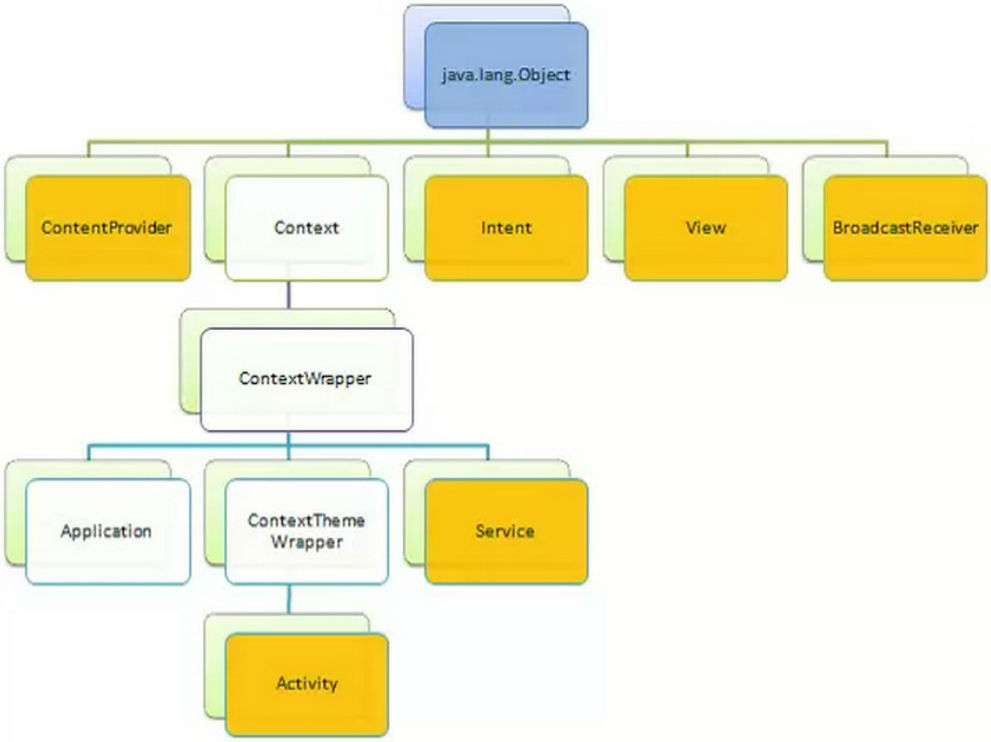

Рис. 2.1. Ієрархія класів Android SDK

На рис. 2.1 показана ієрархія основних класів Android SDK, з якими зазвичай має справу розробник. Насправді класів набагато більше, жовтим кольором виділені класи, з якими розробник працює безпосередньо, успадковує від них свої класи. Інші класи не менш важливі, але вони рідше використовуються безпосередньо. Для початку розглянемо класи Intent і View.

Клас View є основним будівельним блоком для компонентів інтерфейсу користувача(UI), він визначає прямокутну область екрану і відповідає за промальовування і обробку подій. Є базовим класом для віджетів(GUI widgets), які використовуються для створення інтерактивних компонентів для користувача інтерфейсу: кнопок, текстових полів і т.п.

А також є базовим класом для класу ViewGroup, який є невидимим контейнером для інших контейнерів і віджетів, визначає властивості розташування компонентів для користувача інтерфейсу.

Інтерфейс Android-додаток являє собою ієрархію UI компонентів(див. рис. 2.2), можна описати цю ієрархію програмно, але більш простим і ефективним способом задати розташування елементів інтерфейсу є XML файл, який надає зручну для сприйняття структуру компоновки(layout file). Під час виконання XML файл автоматично перетворюється в дерево відповідних об'єктів. Детальніше про клас

View, властивості і методах: http://developer.android.com/reference/android/view/View.html.

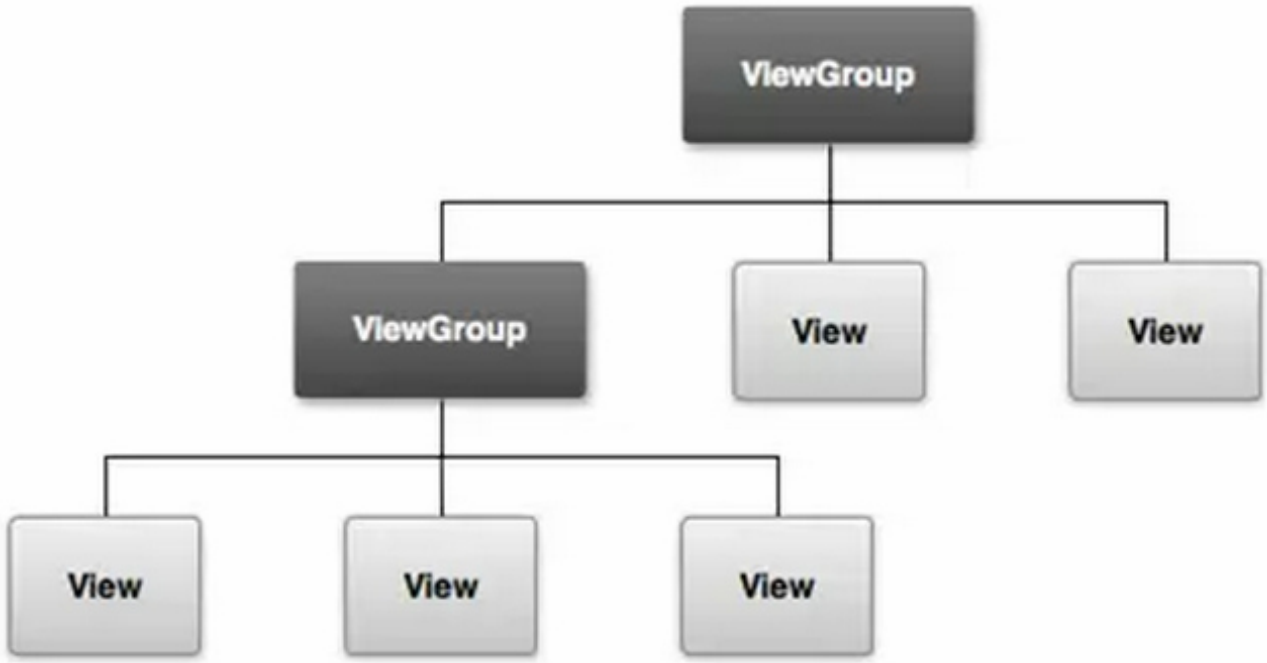

Рис. 2.2. Ієрархія компонентів, що визначає компоновку інтерфейсу користувача

Об'єкти-екземпляри класу Intent використовуються для передачі повідомлень між основними компонентами додатків. Відомо, що три з чотирьох основних компонентів: активності, сервіси та приймачі широкомовних повідомлень, можуть бути активовані за допомогою повідомлень, які називаються намірами. Такі повідомлення є інструментом пізнього зв'язування компонентів одного або декількох додатків. Екземпляр класу Intent являє собою структуру даних, що містить опис операції, яка повинна бути виконана, і зазвичай використовується для запуску активності або сервісу. У випадку з приймачами широкомовних повідомлень об'єкт Intent містить опис події, яка сталася або було оголошено.

Для кожного типу компонентів існують свої механізми передачі намірів.

- Щоб запустити активність або викликати у працюючої активності нову дію, необхідно передати об'єкт-намір в метод Context.startActivity() або Activity.startActivityForResult().
- Щоб запустити сервіс або доставити нові інструкції працюючому сервісу, необхідно передати об'єкт-намір в метод Context.startService(). Також об'єкт-намір

може бути переданий в метод Context.bindService(), щоб зв'язати між собою викликає компонент і сервіс.

- Щоб доставити об'єкт-намір всім зацікавленим приймачів широкомовних повідомлень, необхідно передати його в будь-який з широкомовних методів: Context.sendOrderedBroadcast(), Context.sendStickyBroadcast(), Context.sendBroadcast().

У кожному випадку система Android у відповідь на намір знаходить відповідний компонент: активність, сервіс або безліч широкомовних приймачів і запускає його якщо необхідно. У цій системі повідомлень не трапляється накладок: повідомлення-намір, відправлене певного компоненту, буде отримано саме цим компонентом і ніким іншим.

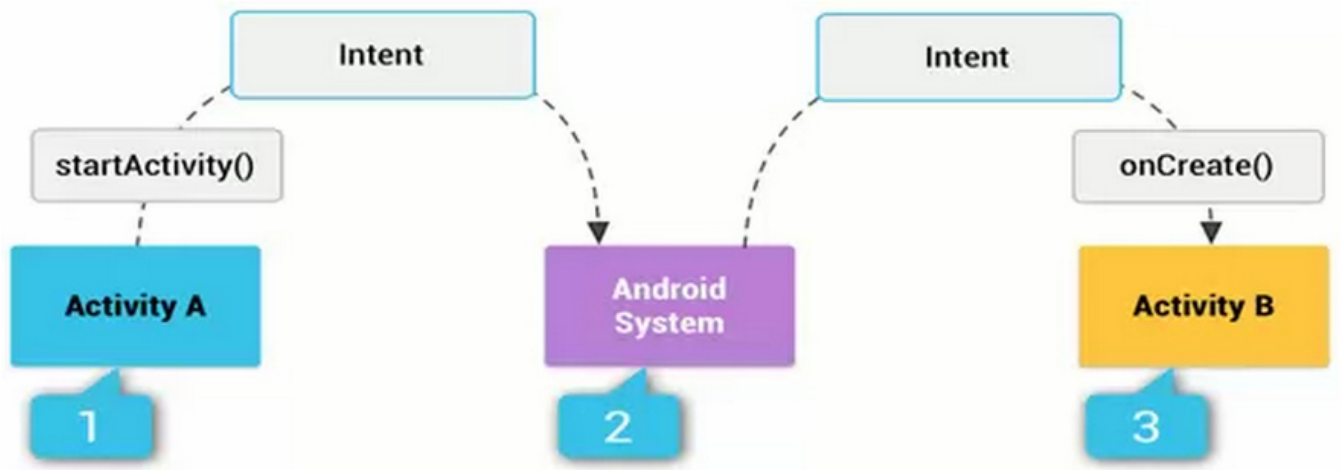

Рис. 2.3. Передача намірів(Intent)

На рис.2.3 можна побачити як відбувається передача намірів(Intent), в даному випадку одна активність запускає іншу. [6] Активність А створює намір(Intent) з описом дії і передає його в метод startActivity(). [7] Система Android перевіряє всі додатки на збіг з наміром, коли збіг знайдено, [8] система запускає відповідну активність, для чого викликає метод onCreate() і передає в нього об'єкт -намереніе Intent.

Детальніше про клас Intent:

http://developer.android.com/guide/components/intents-filters.html ,
http://developer.android.com/reference/android/content/Intent.html

Прийшов час більш серйозно розглянути основні компоненти: активності, сервіси, контент-провайдери, приймачі широкомовних повідомлень. У першу чергу нас буде цікавити життєвий цикл цих компонентів. Що ж таке цей життєвий цикл? Життєвий цикл можна розглядати, як процес функціонування компонента: починаючи з моменту створення і запуску, включаючи активний і неактивний періоди роботи, і, закінчуючи знищенням і звільненням ресурсів.