ЛЕКЦІЯ 4. АКТИВНОСТІ, СЕРВІСИ ТА КОНТЕНТ-ПРОВАЙДЕРИ

АКТИВНОСТІ(ACTIVITIES)

Активність - вікно, що несе графічний інтерфейс користувача. Вікно активності зазвичай займає весь екран пристрою, однак цілком можливо створювати напівпрозорі або плаваючі діалогові вікна. Мобільні додатки зазвичай є багато віконний, тобто. Містять кілька активностей, по одній на кожне вікно. Одна з активностей визначається як "головна", і саме її користувач бачить при першому запуску програми.

Кожен екран додатка є спадкоємцем класу Activity. Для створення активності необхідно створити клас-спадкоємець класу Activity безпосередньо або через будь-якого його нащадка. У цьому класі необхідно реалізувати всі методи, які викликаються системою для управління життєвим циклом активності. Таких методів сім:

onCreate()	- Метод, що викликається системою при створенні активності. У реалізації методу необхідно ініціалізувати основні компоненти активності і в більшості випадків викликати метод setContentView() для підключення відповідного XML-файла компонування(layout file). Після методу onCreate() завжди викликається метод onStart().
onRestart()	- Метод, що викликається системою при необхідності запустити припинену активність. Після цього методу завжди викликається метод onStart().
onStart()	- Метод, що викликається системою безпосередньо перед тим, як активність стане видимою для користувача. Після цього методу викликається onResume().
onResume()	- Метод, що викликається системою безпосередньо перед тим, як активність почне взаємодіяти з користувачем. Після цього методу завжди викликається onPause().
onPause()	- Метод, що викликається системою при втраті активністю фокуса. У цьому методі необхідно фіксувати всі зміни, які повинні бути збережені

	за межами поточної сесії. Після цього методу викликається onResume() , якщо активність повернеться на передній план, або onStop() , якщо активність буде прихована від користувача.
onStop()	- Метод, що викликається системою, коли активність стає невидимою для користувача. Після цього методу викликається або onRestart() , якщо активність повертається до взаємодії з користувачем, або onDestroy() , якщо активність знищується.
onDestroy()	- Метод, що викликається системою перед знищенням активності. Цей метод викликається або коли активність завершується, або коли система знищує активність, щоб звільнити ресурси. Можна розрізняти ці два сценарії за допомогою методу isFinishing(). Це останній виклик, який може прийняти активність.

При реалізації перерахованих вище методів першим ділом завжди необхідно викликати відповідний метод предка. Розглянуті методи визначають життєвий цикл активності. Фактично активність може існувати в одному з трьох станів:

- **Виконується(running).** Активність знаходиться на передньому плані і утримує фокус введення.
- **Призупинена.** Активність частково видима, однак фокус введення втрачений. В цей стан активність потрапляє після виклику методу onPause(). У цьому стані активність підтримується в "бойовій готовності", тобто в будь-який момент може отримати фокус введення і стати активною. Однак в цьому стані процес активності може бути знищений системою, в разі екстремальної браку пам'яті.
- **Зупинена.** Активність повністю невидима. В цей стан активність потрапляє після виклику методу onStop(). У цьому стані активність може бути "викликана до життя", вона зберігає всі стани і необхідну для відновлення інформацію, проте процес активності може бути знищений, якщо пам'ять знадобиться для інших цілей.

СЕРВІСИ(SERVICES)

Сервіс(Service) є компонентом програми, призначеним для виконання тривалих операцій у фоновому режимі. Існує два способи існування сервісів:

- перший полягає в тому, що сервіс запущений(started) і працює самостійно у фоновому режимі, так він може працювати невизначено довго, поки не виконає своє завдання;
- другий полягає в тому, що сервіс прив'язаний(bound) до деякого компоненту або декільком компонентам, в цьому випадку сервіс пропонує інтерфейс для взаємодії з компонентом і працює поки прив'язаний хоча б до одного компоненту, як тільки зв'язок з усіма компонентами розривається сервіс завершує свою роботу.

Для створення сервісу необхідно створити клас-спадкоємець класу Service безпосередньо або через будь-якого його нащадка. При цьому в реалізації класу необхідно перевизначити(тобто. Написати свою реалізацію) деякі методи, керуючі ключовими аспектами життєвого циклу сервісу і забезпечують механізм скріплення компонентів з сервісом, у відповідному випадку. Розглянемо найбільш важливі методи потребують реалізації при створенні сервісу.

onStartCommand()	- Метод, що викликається системою, коли деякий компонент, наприклад активність, викликає метод startService(). У цьому випадку сервіс запускається і може працювати у фоновому режимі невизначено довго, тому необхідно подбати про зупинку сервісу, коли він виконає свою роботу. Для зупинки сервісу використовується метод stopSelf() у випадку, коли сервіс сам припиняє свою роботу, або stopService() у випадку, коли роботу сервісу припиняє деякий компонент. Немає необхідності писати реалізацію методу onStartCommand() , якщо не передбачається самостійної роботи сервісу(тобто він буде працювати тільки в зв'язці з деякими компонентами).
onBind()	- Метод, що викликається системою, коли деякий компонент бажає прив'язати до себе сервіс і викликає метод bindService(). Цей метод повинен повертати реалізацію інтерфейсу IBinder ,

	яка може бути використана компонентом-клієнтом для взаємодії з сервісом. Метод onBind() необхідно реалізувати в будь-якому випадку, але, якщо не передбачається зв'язування сервісу з якими-небудь компонентами, повертається значення повинно бути рівним null.

Необхідно відзначити, що сервіс може бути запущений як самостійна одиниця, а надалі може бути прив'язаний до деяких компонентів. У цьому випадку в сервісі повинні бути обов'язково реалізовані обидва методи onStartCommand() і onBind().

onCreate()	- Метод, що викликається системою, при першому зверненні до сервісу для виконання початкових налаштувань. Цей метод викликається до виклику методів onStartCommand() та / або onBind().
onDestroy()	- Метод, що викликається системою, коли сервіс або виконав всі дії, для яких створювався, або більше не пов'язаний ні з одним компонентом, тобто його послуги більше не потрібні. У реалізації цього методу необхідно передбачити звільнення всіх ресурсів, таких як потоки, зареєстровані слухачі, приймачі і т.д. Виклик цього методу є останніми викликом, який може отримати сервіс.

КОНТЕНТ-ПРОВАЙДЕРИ(CONTENT PROVIDERS)

Контент-провайдер управляє доступом до сховища даних. Для реалізації провайдера в Android додатку має бути створений набір класів відповідно до маніфестом програми. Один з цих класів має бути спадкоємцем класу ContentProvider , який забезпечує інтерфейс між контент-провайдером та іншими додатками.

Основне призначення цього компонента додатка полягає в наданні іншим додаткам доступу до даних, однак ніщо не заважає в додатку мати активність, яка дозволить користувачеві запитувати і змінювати дані, що знаходяться під управлінням контент-провайдера.

У мобільних додатках контент-провайдери необхідні в таких випадках:

- програма надає складні дані або файли інших додатків;
- додаток дозволяє користувачам копіювати складні дані в інші програми;
- програма надає спеціальні варіанти пошуку, використовуючи пошукову платформу(framework).

Якщо додаток вимагає використання контент-провайдера, необхідно виконати кілька етапів для створення цього компонента:

1. Проектування способу зберігання даних. Дані, з якими працюють контент-провайдери, можуть бути організовані двома способами:

- Дані представлені файлом, наприклад, фотографії, аудіо або відео. У цьому випадку необхідно зберігати дані у власній області пам'яті програми. У відповідь на запит від іншої програми, провайдер може повертати посилання на файл.
- Дані представлені деякою структурою, наприклад, таблиця, масив. У цьому випадку необхідно зберігати дані в табличній формі. Рядок таблиці являє собою деяку сутність, наприклад, співробітник або товар. А стовпець - деяка властивість цієї сутності, наприклад, ім'я співробітника або ціна товару. В системі Android загальний спосіб зберігання подібних даних - база даних SQLite, але можна використовувати будь-який спосіб постійного зберігання.

2. Створення класу-спадкоємця від класу ContentProvider безпосередньо або через будь-якого його нащадка. При цьому в реалізації класу необхідно перевизначити(тобто написати свою реалізацію) обов'язкові методи.

query()	- Метод, який видобуває дані з провайдера, в якості аргументів отримує таблицю, рядки і стовпці, а також порядок сортування результату, повертає об'єкт типу Cursor.
insert()	- Метод, який додає новий рядок, в якості аргументів отримує таблицю, і значення елементів рядка, повертає URI доданої рядка.
update()	- Метод, оновлюючий існуючі рядки, в якості аргументів отримує таблицю, рядки для оновлення і нові значення елементів рядків, повертає кількість оновлених рядків.
delete()	- Метод, що видаляє рядки, в якості аргументів приймає таблицю і рядки для видалення, повертає кількість вилучених рядків.

getType()	- Метод, який повертає String у форматі MIME, який описує тип даних, відповідний URI.
onCreate()	- Метод, що викликається системою, відразу після створення провайдера, включає ініціалізацію провайдера. Варто відзначити, що провайдер не створюється до тих пір, поки об'єкт ContentResolver чи не спробує отримати до нього доступ.

Створений контент-провайдер управляє доступом до структурованих даних, виконуючи обробку запитів від інших додатків. Всі запити, зрештою, викликають об'єкт ContentResolver , який в свою чергу викликає відповідний метод об'єкта ContentProvider для отримання доступу. Всі перераховані вище методи, крім onCreate() , викликаються додатком-клієнтом. І всі ці методи мають таку ж сигнатуру, як однойменні методи класу ContentResolver. Детальніше про

3. Визначення рядки авторизації провайдера, URI для його рядків та імен стовпців. Якщо від провайдера потрібно керувати намірами, необхідно визначити дії намірів, зовнішні дані і прапори. Також необхідно визначити дозволи, які необхідні додаткам для доступу до даних провайдера. Всі ці значення необхідно визначити як константи в окремому класі, цей клас надалі можна надати іншим розробникам.

Детальніше:

http://developer.android.com/guide/topics/providers/content-provider-creating.html#ContentURI

Детальніше про наміри:

http://developer.android.com/guide/topics/providers/content-provider-creating.html#Intents