Тема 9. Служби і сервіси мобільних платформ

Програмний компонент: Служби (Services). Service є компонентом програми, який може виконувати тривалі операції у фоновому режимі і який не містить користувацького інтерфейсу. Інший компонент програми може запустити службу, яка продовжить роботу у фоновому режимі навіть у тому випадку, коли користувач перейде в інший додаток. Крім того, компонент може прив’язатися до служби для взаємодії з нею і навіть виконувати міжпроцесну взаємодію (IPC). Наприклад, служба може обробляти мережні транзакції, відтворювати музику, виконувати введення-виведення файлу або взаємодіяти з постачальником контенту, і все це у фоновому режимі.

Фактично служба може прибирати дві форми:

1. Запущену. Служба є «запущеною», коли компонент додатка (наприклад, операція) запускає її викликом startService(). Після запуску служба може працювати у фоновому режимі протягом необмеженого часу, навіть якщо був знищений компонент, який її запустив. Зазвичай запущена служба виконує одну операцію і не повертає результатів викликаючому компоненту. Коли операція виконана, служба має зупинитися самостійно.

2. Прив’язану. Служба є «прив’язаною», коли компонент програми прив’язується до неї викликом bindService(). Прив’язана служба пропонує інтерфейс клієнт-сервер, який дозволяє компонентам взаємодіяти зі службою, відправляти запити, отримувати результати і навіть робити це між різними процесами за допомогою міжпроцесної взаємодії (IPC). Прив’язана служба працює до тих пір, поки до неї прив’язаний інший компонент програми. До служби можуть бути прив’язані кілька функцій одночасно, але коли вони скасовують прив’язку, служба знищується.

Незалежно від стану програми (запущена, прив’язана або обидва відразу) будь-який компонент програми може використовувати службу (навіть з окремого додатка) подібно до того, як будь-який компонент може використовувати операцію – запустивши її з допомогою Intent. Однак можна

оголосити закриту службу у файлі маніфесту і заблокувати доступ до неї з інших додатків.

Щоб створити службу, необхідно створити підклас класу Service (або одного з існуючих його підкласів). У вашій реалізації необхідно перевизначити деякі методи зворотного виклику, які обробляють ключові моменти життєвого циклу служби та за необхідності надають механізм прив’язування компонентів.

Найбільш важливими методами зворотного виклику, які необхідно перевизначити, є:

– onStartCommand(). Система викликає цей метод, коли інший компонент, наприклад, операція, запитує запуск цієї служби, викликаючи startService(). Після виконання цього методу служба запускається і може протягом необмеженого часу працювати у фоновому режимі. Якщо ви релізуєте такий метод, то необхідно зупинити службу за допомогою виклику stopSelf() або stopService(). (Якщо потрібно тільки забезпечити прив’язку, реалізовувати цей метод не обов’язково).

– onBind(). Система викликає цей метод, коли інший компонент хоче виконати прив’язку до служби (наприклад, для виконання віддаленого виклику процедури) шляхом виклику bindService(). У вашій реалізації цього методу необхідно забезпечити інтерфейс, який клієнти використовують для взаємодії зі службою, повертаючи IBinder. Завжди необхідно реалізовувати цей метод, але якщо прив’язка непотрібна, необхідно повертати значення null.

– onCreate(). Система викликає цей метод при першому створенні служби для виконання одноразових процедур налаштування (перед викликом onStartCommand() або onBind()). Якщо служба вже запущена, цей метод не викликається.

– onDestroy(). Система викликає цей метод, коли служба більше не використовується та коли виконується її знищення. Ваша служба повинна реалізувати це для очищення ресурсів, таких, як потоки, зареєстровані приймачі, ресивери і т. д. Це останній виклик, який отримує служба.

Якщо компонент запускає службу за допомогою виклику startService() (що приводить до виклику onStartCommand()), то служба продовжує роботу, поки вона не зупиниться самостійно з допомогою stopSelf() або поки інший компонент не зупинить її за допомогою виклику stopService().

Якщо компонент викликає bindService() для створення служби (і onStartCommand() не викликається), то служба працює, поки до неї прив'язаний компонент. Як тільки виконується скасування прив'язки служби до всіх клієнтів, система знищує службу.

Система Android буде примусово зупиняти службу тільки в тому випадку, коли не вистачає пам'яті і коли необхідно відновити системні операції, які відображаються на передньому плані. Якщо служба прив'язана до операції, яка відображається на передньому плані, менш ймовірно, що вона буде знищена, і якщо служба оголошена для виконання у фоновому режимі (як обговорювалося вище), вона майже ніколи не буде знищуватися. В іншому випадку, якщо служба була запущена і є тривалою, система з часом буде опускати її положення в списку фонових завдань, і служба стане дуже чутливою до знищення – якщо ваша служба працює, то ви повинні передбачити витончену обробку її перезапуску системою. Якщо система знищує вашу службу, вона запускає її, як тільки знову з'являється доступ до ресурсів.

Служба – це просто компонент, який може виконуватися у фоновому режимі, навіть коли користувач не взаємодіє з додатком. Отже, необхідно створювати службу тільки в тому випадку, якщо вам потрібно саме це.

Якщо вам потрібно виконати роботу за межами основного потоку, але тільки тоді, коли користувач взаємодіє з додатком, то вам, ймовірно, слід створити новий потік, а не службу. Наприклад, якщо ви бажаєте відтворювати певну музику, але тільки під час роботи, операції, то необхідно створити потік в onCreate(), запустити його виконання в методі onStart(), а потім зупинити його в методі onStop(). Також розгляньте можливість використання класу AsyncTask або HandlerThread замість звичайного класу Thread. У документі «Процеси і потоки» міститься додаткова інформація про ці потоки.

Пам’ятайте, що якщо ви використовуєте службу, вона виконується в основному потоці вашого додатка за замовчуванням, тому потрібно створити новий потік в службі, якщо вона виконує інтенсивні або блокувальні операції.

Запущена служба – це служба, яку запускає інший компонент викликом startService(), що приводить до виклику методу onStartCommand() служби.

При запуску служба має термін життя, що не залежить від компонента, що її запустив, і може працювати у фоновому режимі протягом необмеженого часу, навіть якщо був знищений компонент, який її запустив. Тому після виконання своєї роботи служба повинна зупинитися самостійно за допомогою виклику методу stopSelf(), або її може зупинити інший компонент за допомогою виклику методу stopService().

Компонент програми, наприклад операція, може запустити службу, викликавши метод startService() і передавши об’єкт Intent, який вказує службу і будь-які дані, які служба повинна використовувати. Служба отримує цей об’єкт Intent в методі onStartCommand().

Припустимо, що операції потрібно зберегти певні дані в мережній базі даних. Операція може запустити службу та надати їй дані для збереження, передавши намір метод startService(). Служба отримує намір в методі onStartCommand(), підключається до Інтернету і виконує транзакцію з базою даних. Коли транзакція виконана, служба зупиняється самостійно і знищується.

Традиційно є два класи, які можна успадкувати для створення запущеної служби:

– Service. Це базовий клас для всіх служб. Коли наслідується цей клас, важливо створити новий потік, в якому буде виконуватися вся робота служби, оскільки за замовчуванням служба використовує основний потік вашого додатка, що може уповільнити будь-яку операцію, яку виконує ваш додаток.

– IntentService. Цей підклас класу Service використовує робочий потік для обробки всіх запитів запуску по черзі. Це оптимальний варіант, якщо вам не потрібно, щоб ваша служба обробляла декілька запитів одночасно. Достатньо

реалізувати метод onHandleIntent(), який отримує намір для кожного запиту запуску, дозволяючи виконувати фонову роботу.

У наступних розділах описано, як реалізувати службу з допомогою будь-якого з цих класів.

Через те що більшості запущених додатків не потрібно обробляти декілька запитів одночасно (що може бути дійсно небезпечним сценарієм), ймовірно, буде краще, якщо реалізуєте свою службу з допомогою класу IntentService.

Клас IntentService виконує наступне:

1. Створює робочий потік за замовчуванням, який виконує всі наміри, доставлені в метод onStartCommand(), окремо від основного потоку вашого додатка.

2. Створює робочу чергу, яка передає наміри по одному в вашу реалізацію методу onHandleIntent(), тому не треба турбуватися щодо багатопоточності.

3. Зупиняє службу після обробки всіх запитів запуску, тому вам ніколи не потрібно викликати stopSelf().

4. Здійснює реалізацію методу onBind(), яка повертає null.

5. Здійснює реалізацію методу onStartCommand() за замовчуванням, яка відправляє намір в робочу чергу і потім у вашу реалізацію onHandleIntent().

Все це означає, що вам достатньо реалізувати метод onHandleIntent() для виконання роботи, наданої клієнтом (хоча, крім того, ви повинні надати маленький конструктор для служби).

Все, що потрібно: конструктор і реалізація класу onHandleIntent().

Якщо необхідно перевизначити також і інші методи зворотного виклику, такі, як onCreate(), onStartCommand() або onDestroy(), обов’язково викличте реалізацію суперкласу, щоб клас IntentService міг правильно обробляти життєвий цикл робочого потоку.

Запущена служба повинна керувати своїм життєвим циклом. Тобто система не зупиняє і не знищує службу, якщо не потрібно відновити пам’ять системи, і служба продовжує роботу після повернення з методу onStartCommand(). Тому служба повинна зупинятися самостійно за допомогою

виклику методу stopSelf(), або інший компонент може зупинити її за допомогою виклику методу stopService().

Отримавши запит на зупинку за допомогою stopSelf() або stopService(), система якомога швидше знищує службу.

Однак якщо служба обробляє кілька запитів onStartCommand() одночасно, то не треба зупиняти службу після завершення обробки запиту запуску, оскільки, ймовірно, вже був отриманий новий запит запуску (зупинка в кінці першого запиту призвела б до переривання другого). Щоб уникнути цієї проблеми, можна використовувати метод stopSelf(int), який гарантує, що ваш запит на зупинку служби завжди заснований на самому останньому запиті запуску. Тобто коли ви викликаєте stopSelf(int), передається ідентифікатор запиту запуску (ідентифікатор startId, доставлений в onStartCommand()), якому відповідає ваш запит зупинки.

Прив’язана служба – це служба, яка допускає прив’язку до неї компонентів програми за допомогою виклику bindService() для створення довготривалого з’єднання (і зазвичай не дозволяє компонентам запускати її за допомогою виклику startService()).

Прив’язана служба створюється, коли треба взаємодіяти зі службою операцій та з іншими компонентами вашого додатка чи показувати деякі функції вашого додатка іншим додаткам за допомогою міжпроцесної взаємодії (IPC).

Щоб створити прив’язану службу, необхідно реалізувати метод зворотного виклику onBind() для повернення об’єкта IBinder, який визначає інтерфейс взаємодії зі службою. Після цього інші компоненти програми можуть ви кликати метод bindService() для вилучення інтерфейсу і почати викликати методи служби. Служба існує тільки для обслуговування прив’язаного до неї компонента, а тому, коли немає компонентів, прив’язаних до служби, система знищує її (вам не потрібно зупиняти прив’язану службу, як це вимагається для служби, запущеної за допомогою onStartCommand()).

Щоб створити прив’язану службу, необхідно в першу чергу визначити інтерфейс взаємодії клієнта зі службою. Цей інтерфейс між службою та клієнтом

повинен бути реалізацією об’єкта IBinder, яку ваша служба повинна повертати з методу зворотного виклику onBind(). Після того як клієнт отримає об’єкт IBinder, він може почати взаємодію зі службою за допомогою цього інтерфейсу.

Одночасно до служби можуть бути прив’язані кілька клієнтів. Коли клієнт закінчує взаємодію зі службою, він викликає unbindService() для скасування прив’язки. Як тільки не залишається ні одного клієнта, прив’язаного до служби, система знищує службу.

Після запуску служба може повідомляти користувача про події, використовуючи спливаючі попередження або повідомлення в рядку стану.

Спливаюче повідомлення – це повідомлення, що короткочасно з’являється на поточному вікні, тоді як повідомлення в рядку стану – це значок в рядку стану з повідомленням, що користувач може вибрати, щоб виконати дію (таку, як запуск операції).

Зазвичай повідомлення в рядку стану є найбільш зручним рішенням, коли завершується якась фонова робота (наприклад, завершено завантаження файлу), і користувач може діяти. Коли користувач вибирає повідомлення в розширеному вигляді, повідомлення може запустити операцію (наприклад, для перегляду завантаженого файлу).

Служба переднього плану – це служба, про яку користувач активно обізнаний, і тому вона не є кандидатом для видалення системою в разі нестачі пам’яті. Служба переднього плану повинна виводити повідомлення в рядок стану, який знаходиться під заголовком «Постійні». Це означає, що повідомлення не може бути видалене, поки служба не буде зупинена або видалена з переднього плану.

Наприклад, музичний програвач, який відтворює музику зі служби, має бути налаштований на роботу на передньому плані, тому що користувач точно знає про його роботу. Повідомлення в рядку стану може показувати поточну музичну композицію і дозволяти користувачу запускати операцію для взаємодії з музичним програвачем.

Життєвий цикл служби набагато простіший, ніж життєвий цикл операції. Однак набагато важливіше приділити пильну увагу тому, як ваша служба створюється і знищується, тому що служба може працювати у фоновому режимі без відома користувача.

Життєвий цикл служби, від створення до знищення, може слідувати двома різними шляхами, а саме:

1. Запущена служба. Служба створюється, коли інший компонент викликає метод startService(). Потім служба працює протягом необмеженого часу і має зупинитися самостійно за допомогою виклику методу stopSelf(). Інший компонент також може зупинити службу допомогою виклику методу stopService(). Коли служба зупиняється, система знищує її.

2. Прив’язана служба. Служба створюється, коли інший компонент (клієнт) викликає метод bindService(). Потім клієнт взаємодіє зі службою через інтерфейс IBinder. Клієнт може закрити з’єднання за допомогою виклику методу unbindService(). До однієї служби можуть бути прив’язані кілька клієнтів, і коли вони скасовують прив’язку, система знищує службу (служба не повинна зупинятися самостійно).

Ці дві служби необов’язково працюють незалежно одна від одної. Тобто можна прив’язати службу, яка вже була запущена за допомогою методу startService(). Наприклад, фонова музична служба може бути запущена за допомогою виклику методу startService() з об’єктом Intent, який ідентифікує музику для відтворення. Пізніше, наприклад, коли користувач хоче отримати доступ до управління програвачем, операція може встановити прив’язку до служби за допомогою виклику методу bindService(). У подібних випадках методи stopService() і stopSelf() фактично не зупиняють службу, поки не буде скасована прив’язка всіх клієнтів.

Прив’язана служба надає інтерфейс типу клієнт-сервер. Прив’язана служба дозволяє компонентам додатка (наприклад, операціям) взаємодіяти зі службою, відправляти запити, отримувати результати і навіть робити те ж саме з іншими процесами через IPC. Прив’язана служба зазвичай працює, поки інший

компонент додатка прив’язаний до неї. Вона не працює постійно у фоновому режимі.

Прив’язана служба являє собою реалізацію класу Service, яка дозволяє іншим програмам прив’язуватися до нього і взаємодіяти з ним. Щоб забезпечити прив’язку служби, спочатку необхідно реалізувати метод зворотного виклику onBind(). Цей метод повертає об’єкт IBinder. Він визначає програмний інтерфейс, за допомогою якого клієнти можуть взаємодіяти зі службою.