Введение в разработку под Android

Семинар 6

На этом семинаре

- Сервисы и нюансы работы с ними
- Примеры работы с сервисами

Сервисы

https://developer.android.com/guide/components/services

Компонент приложения, который позволяет выполнять “долгие” операции в фоне.

Основные особенности:

- не имеет пользовательского интерфейса
- продолжает выполнение даже если пользователь свернул приложение
- другой компонент может быть связан с сервисом для взаимодействия (допускается IPC)

Примеры задач, которые решаются с помощью сервисов:

- синхронизации на файловой системе
- проигрывание музыки
- загрузки файлов по сети
- …

Замечание: сервис исполняется в главном потоке родительского процесса; сервис не создает своего собственного потока или процесса (если это не указано явно). Все блокирующие операции должны быть вынесены в отдельный поток, чтобы избежать **ANR**.

Типы сервисов

https://developer.android.com/guide/components/services

Foreground

Для операций заметных пользователю. Обязан показывать пользователю нотификацию. Эта нотификация не может быть скрыта пока сервис не завершит свою работу. Пример: проигрыватель музыки.

Background

Для операций необходимых вашему приложению и не требующих внимания пользователя. Пример: синхронизации данных на файловой системе.

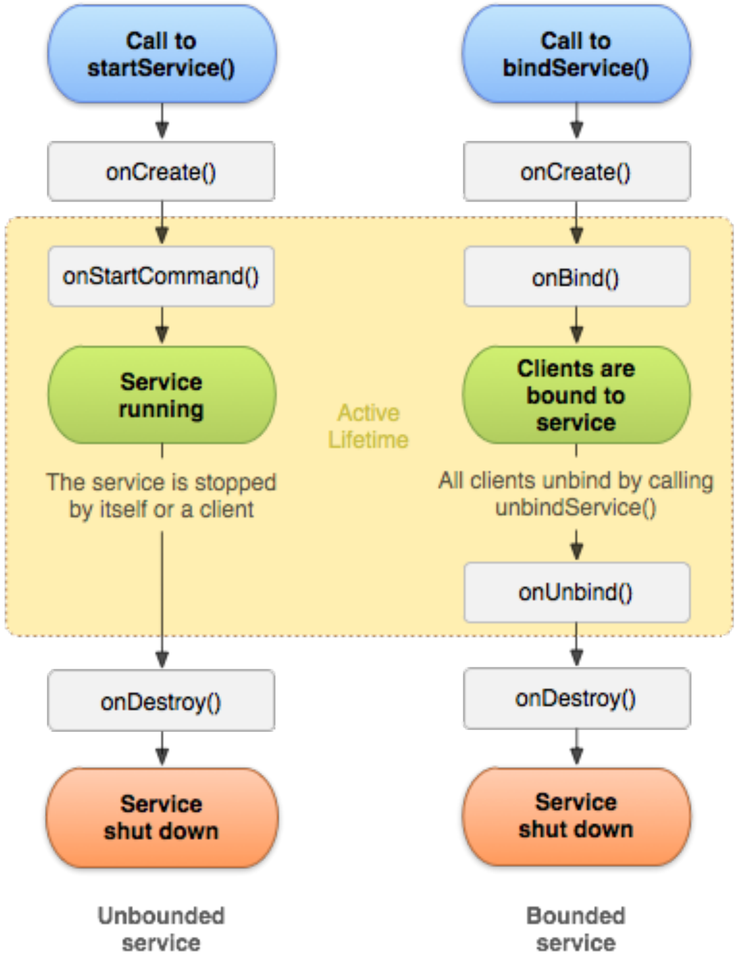
Bounded

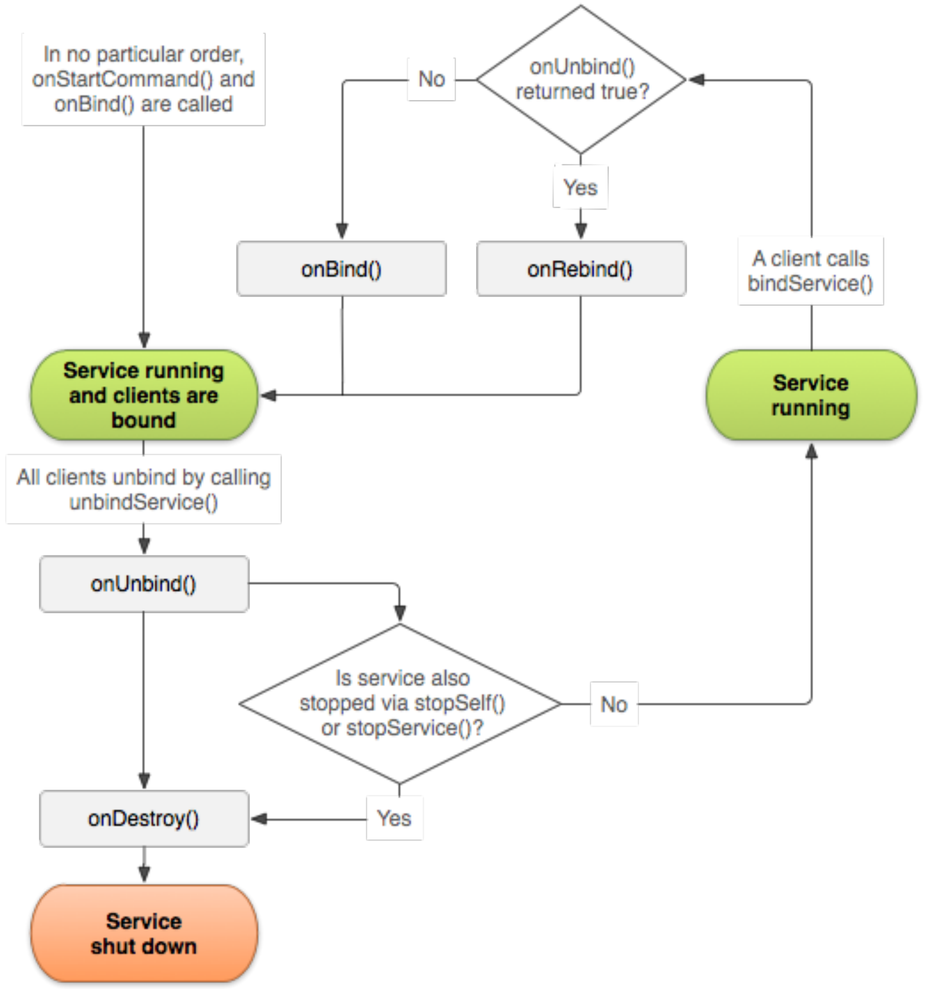
Дает пользователю интерфейс вида клиент-сервер. Позволяет другим компонентам взаимодействовать с сервисом по средствам этого интерфейса (в том числе нескольким сразу).

Замечание: WorkManager - альтернатива сервисам, при условии, что полезная работа не должна выполняться пока ваше приложение свернуто.

Lifecycle сервисов

https://developer.android.com/guide/components/services





Создание сервисов

https://developer.android.com/guide/components/services

В файле манифеста:

```
<manifest>
<application ... >
<service android:name=".ExampleService" />
</application>
</manifest>
```

В коде:

```
Intent intent = new Intent(...);
startService(intent);
```

Замечание: для того, чтобы обеспечить безопасность ваших приложение всегда запускайте сервис с помощью explicit intent, не используйте intent-filters.

Замечания про onStartCommand

https://developer.android.com/guide/components/services

Метод **onStartCommand** возвращает код, который определяет поведение сервиса в случае разрушения сервиса.

START_NOT_STICKY

Не создаем сервис повторно. Это самый безопасный вариант, позволяющий избежать запуска вашего сервиса, когда в нем нет необходимости, и когда ваше приложение может просто перезапустить задачи.

START_STICKY

Пересоздаем сервис и вызываем onStartCommand (игнорируя последний intent). Это подходит для медиаплееров (или подобных им сервисов), которые не выполняют команды, а работают неопределенно долго и ждут следующей задачи.

START_REDELIVER_INTENT

Пересоздаем сервис и вызываем onStartCommand (с последним intent, который был доставлен). Все остальные intent’ы доставляются по очереди. Это подходит для сервисов, активно выполняющих работу, которую следует немедленно возобновить, например загрузка файла.

Остановка сервисов

https://developer.android.com/guide/components/services

Запущенный сервис **сам управляет своим жизненным циклом**. Кроме случаев, когда системе требуется освободить ресурсы.

Чтобы остановить сервис:

- из сервиса stopSelf
- из другого компонента stopService
- **система разрушает сервис асинхронно**

Замечание: для того, чтобы не потреблять лишние ресурсы (в том числе батарею) ваше приложение должно завершать сервисы, которые не выполняют полезной работы.

Дополнительно можно

https://developer.android.com/courses/fundamentals-training

- JobIntentService, Messenger и др.
- AIDL
- Расширения Binder