UNITÉ D ’ENSEIGNEMENT (UE) : DÉVELOPPEMENT MOBILE

Ch5 : Services, Broadcast Receivers, Content Providers

BROADCAST RECEIVERS

- **Un Broadcast Receiver est un composant qui répond à des annonces diffusées pour tout le système**
- **Plusieurs broadcasts sont originaires du système, par ex:**
  - Annonce que l'écran est éteint
  - Le niveau de la batterie est bas
  - Une photo a été prise...
- **D'autres sont initiés par des applications**
  - Par ex, annoncer à une autre application que des données ont été téléchargées et disponibles pour être utilisées
- **N'ont pas d'interface, mais...**
  - Peuvent afficher une notification dans la barre de statut pour alerter l'utilisateur

Broadcast Intent

- **Intent envoyé à toutes les applications qui ont souscrit à un Broadcast Receiver**
- **Envoyé par le système Android, par exemple, pour indiquer les changements de l'appareil**
  - Démarrage terminé,
  - connexion à un appareil externe
  - Écran allumé/éteint
- **Peut être:**
  - Normal (ou asynchrone): envoyé à tous les Broadcast Receivers en même temps
  - Ordonné: envoyé à un receiver, qui peut, soit le tuer, soit le passer à un autre Broadcast Receiver

Broadcast Receiver

- **Mécanisme avec lequel les applications peuvent répondre aux *Broadcast Intents***
- **Doit être souscrit par une application, et configuré dans un Intent Filter, pour indiquer le type de broadcast auquel elle est intéressée.**
- **Quand un intent correspondant est diffusé**
  - Le Broadcast Receiver est invoqué par l'environnement d'exécution Android, même si l'application intéressée n'est pas démarrée
  - Le Broadcast Receiver a 5 secondes pour compléter les tâches qu'il a à faire
    - Lancement d'un service, mise à jour de données ou notification à un utilisateur
- **S'exécute en arrière plan et n'a pas d'interface utilisateur**

Code: Broadcast Intent

- **Pour créer un Broadcast Intent, il faut:**
  - Créer dans le listener de l'évènement qui va déclencher le broadcast:
    - Un nouvel intent i
    - Définir une action pour cet intent avec la méthode i.setAction
    - Déclencher le broadcast avec la méthode sendBroadcast(i)

Code: Broadcast Receiver

- **Pour créer un Broadcast Receiver, il faut:**
  - Créer un nouveau projet sans Activité
    - Dans votre projet, choisir New -> Other -> Broadcast Receiver, Vous remarquerez que:
      - Une nouvelle classe étendant BroadcastReceiver a été créée
      - Un nouvel élément receiver a été rajouté à votre fichier Manifest
  - Implémenter la méthode onReceive de votre receiver, qui indiquera le comportement à adopter à la réception de l'évènement diffusé
  - Ajouter dans le fichier Manifest pour ce receiver, sous l'élément « intent filters », l'action que vous avez précédemment associé au Broadcast Intent

Les actions

- **android.intent.action.BATTERY_LOW**
- **android.intent.action.BOOT_COMPLETED**
- **android.intent.action.CALL**
- **android.intent.action.DATE_CHANGED**
- **android.intent.action.REBOOT**
- **android.net.conn.CONNECTIVITY_CHANGE**
- **android.intent.action.AIRPLANE_MODE**

Content Provider

- **Gère l’accès à des données structurées**
- **Implémente un mécanisme pour le partage de données entre applications**
- **Toute application peut fournir aux autres applications un accès à ses données sous-jacentes via l’implémentation d’un Content Provider permettant d’ajouter, supprimer et lancer des requêtes sur les données**
- **Définit des URI pour l’accès aux données**
- **Données peuvent être partagées sous forme de fichier ou de base de données SQLite**
- **Existence de Content Providers standards dans le runtime Android pour accès aux données: Contacts ou Fichiers Média**

Quand implémenter un Content Provider?

- **Vous avez besoin d’un Content Provider quand vous voulez:**
  - Offrir des données complexes ou des fichiers à d’autres applications
  - Permettre aux utilisateurs de copier des données complexes vers d’autres applications
  - Fournir des suggestions de recherche pour votre application en utilisant le framework de recherche
- **Vous n’avez pas besoin d’un Content Provider :**
  - Pour utiliser une base de données SQLite, si l’utilisation est entièrement interne à votre application

Etapes

- **Concevoir votre espace de stockage pour vos données. Il peut être sous la forme de:**
  - Fichiers
    - Photos, audio ou vidéos
    - Les stocker dans l'espace privé de votre application
  - Données Structurées
    - Données dans des bases de données, tableaux…
    - Stocker les données dans une forme compatible avec des tables de lignes/colonnes
- **Définir une implémentation concrète du ContentProvider et ses méthodes requises**
- **Définir les URIs, les noms de colonnes du provider, les intent actions, les permissions...**

Code

- **Pour accéder aux données d'un content provider, utiliser un *ContentResolver* dans le contexte de votre application**
  - Communique avec l'objet Provider (une instance de la classe implémentant ContentProvider)
  - L'objet provider reçoit des requêtes des clients, les exécute et retourne le résultat
- **Implémenter les méthodes suivantes de ContentProvider**
  - query(): extraction de données du provider
  - insert(): insertion d'une nouvelle ligne dans votre provider
  - update(): modification d'une ligne existante de votre provider
  - delete(): suppression d'une ligne de votre provider
  - getType() : retourne le type MIME correspondant à un URI
  - onCreate(): initialisation du provider

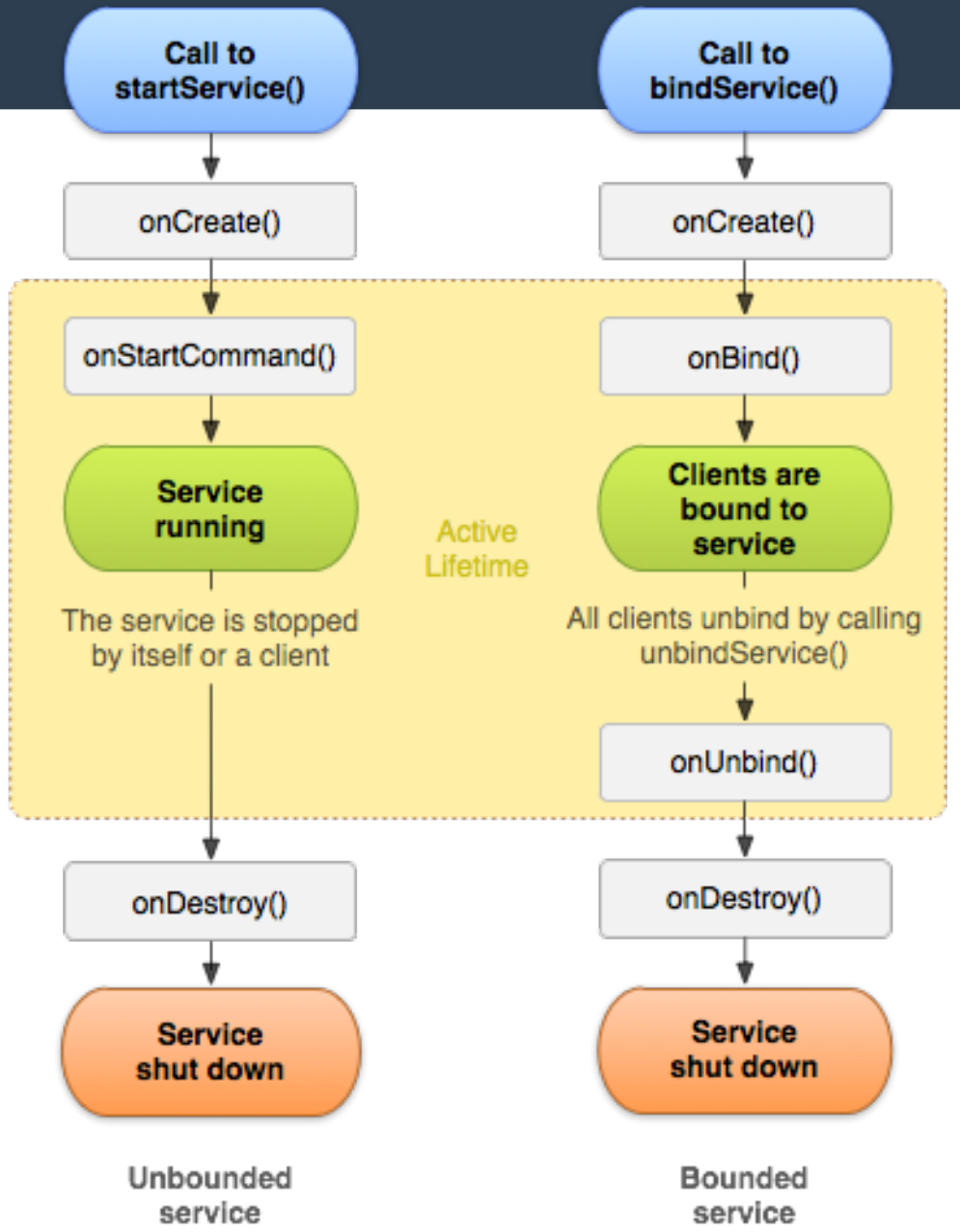
SERVICES

- **Processus qui tourne en arrière plan et n'a pas d'interface**
- **Peut être démarré et géré à partir d'activités, de Broadcast Receivers ou autres services**
- **Idéal pour des situations où une application a besoin de continuer à réaliser des tâches sans avoir besoin d'une interface visible pour l'utilisateur**
- **Peut notifier les utilisateurs d'évènements grâce aux notifications et toasts**
- **Peut lancer des Intents**
- **Services ont une plus forte priorité que les autres processus et sont terminés en dernier par le système, s'il a besoin de ressources**
  - Le service sera redémarré automatiquement dès que les ressources nécessaires sont disponibles à nouveau

Types de Services

- **Deux utilisations de services**
  - Local vs global
  - Started vs Bound
- **Cycle de vie spécifique aux services (quelque soit leur type)**
- **Services « Locaux »**
  - Service personnel, inaccessible pour les autres applications
  - Service, Intent, IntentService
    - En accès restreint
  - Peu utilisés en pratique
- **Services « Distants »**
  - Accessible aux autres applications
  - Langage commun AIDL pour définir les interfaces
  - Recherche du service, IBinder , Connection

Cycle de Vie d'un Service



Services « Started »

- **Démarré par un autre composant de l'application (ex Activity) suite à l'appel de startService() et implémente onStartCommand()**
- **Une fois démarré, il peut s'exécuter indéfiniment en arrière plan**
  - Même si le composant qui l'a démarré est détruit
- **En général : Réalise une seule opération et ne retourne pas de résultat au processus qui l'a déclenché**
- **Le service doit se terminer explicitement:**
  - Soit lui-même : appel à stopSelf()
  - Soit par une application cliente : appel à stopService()
- **Exemple : service de téléchargement / upload de fichiers**

Services « Bound »

- **Un composant de l’application se connecte à ce service en appelant bindService()**
- **Un service Bound offre une interface qui permet à ses clients d’interagir avec lui**
  - Envoyer des requêtes / Recevoir des réponses
  - Même s’il est sur un autre processus
- **S ’exécute uniquement quand un autre composant de l’application est connecté à lui**
- **Plusieurs composants peuvent se connecter au service en même temps**
- **Le service est détruit quand tous les composants se déconnectent**
  - Destruction automatique par le système Android
- **Exemple:**
  - Lecture d’un fichier MP3, et commande d’arrêt/pause/replay… via l’interface

Services « Mixtes »

- **Un service peut être à la fois Started et Bound**
- **Started**
  - S’exécute indéfiniment
  - Implémente la méthode onStartCommand()
- **Bound**
  - Est relié à un composant (activité ou autre)
  - Implémente la méthode onBind()
- **Dans tous les cas (Started , Bound ou Mixte)**
  - Par défaut, tout composant, même à partir d’une autre application, peut lancer un service grâce aux Intents
  - Il est possible de déclarer un service comme étant Privé à l’application : Dans le Manifest

Intent Service

- **Classe utilitaire : Sous-classe de Service**
- **Lance un « worker thread » pour gérer les tâches en arrière plan de manière asynchrone**
- **Les requêtes sont stockées dans des queues (files d'attente) et traitées séquentiellement (dans leur ordre d'arrivée)**
- **Une opération lancée dans un IntentService ne peut pas être interrompue**
- **Quand toutes les requêtes sont exécutées, le thread termine**
- **On doit implémenter la méthode onHandleIntent**
  - Code à exécuter pour chaque requête

Services et Threads

- **Par défaut, un service s'exécute dans le thread principal du processus qui le contient**
- **Si le service réalise des opérations de calcul intensif ou des opérations bloquantes (lecture mp3, communication réseau) le développeur doit (explicitement) créer un nouveau thread pour le service pour réaliser ces tâches**
  - Sauf dans le cas d'un IntentService , qui tourne par défaut sur son propre worker thread
- **Objectifs :**
  - Garder le thread principal réactif vis à vis des interactions avec l'utilisateur
  - Réduire le risque du problème : Application Not Responding (ANR)

Services ou Threads ?

- **Un service est simplement un composant qui peut fonctionner en arrière-plan même si l'utilisateur n'interagit pas avec votre application.**
  - On doit créer un service uniquement pour cet objectif
- **Si le besoin est d'effectuer un travail en dehors du thread principal, mais seulement pendant que l'utilisateur interagit avec l'application -> Il est recommandé de créer un nouveau thread et non pas un service**
- **Exemple (jouer de la musique uniquement quand votre activité s'exécute):**
  - Créer un thread dans onCreate ()
  - Commencer à l'exécuter dans onStart ()
  - Puis l'a rrêter dans onStop ().
- **Un service fonctionne par défaut dans le thread principal de l'application**
  - Il faut toujours créer un nouveau thread dans le service s'il effectue des opérations intensives ou bloquantes.