



# Developpement mobile – Services

AKKA ZEMMARI

ZEMMARI@LABRI.FR



# Les services

- ▶ Un service est une activité sans interface graphique
- ▶ Sert à effectuer des opérations ou des calculs en dehors de l'interaction utilisateur
  - ▶ ATTENTION : même si le Service s'exécute de façon invisible, il s'exécute dans le **thread principal** de l'application. Il faut qu'il démarre un thread secondaire si on veut récupérer la main.
- ▶ Exemples d'utilisation
  - ▶ J'écoute un fichier audio mp3, pendant que je navigue sur le web,
  - ▶ Dès que je m'empare du mobile, une sonnerie retentit,
  - ▶ Les coordonnées GPS sont envoyées régulièrement,
  - ▶ Dès que je m'approche d'une zone dangereuse mon mobile s'éteint,



# Les services

- ▶ <http://developer.android.com/reference/android/app/Service.html>
- ▶ Un service est une classe qui étend `android.app.Service`
- ▶ Deux architectures de services
  - ▶ **Service local** : services qui s'exécutent dans le même processus que votre application
  - ▶ **Service « distant »** : services qui s'exécutent dans des processus indépendants de votre application
- ▶ Deux modes de liaison (valables pour les 2 architectures)
  - ▶ **Unbounded** : permet de démarrer et de stopper le Service
  - ▶ **Bounded** : permet en plus à l'activité d'appeler des méthodes locale du Service



# Démarrage et arrêt d'un Service

## ▶ Mode *Unbounded*

- ▶ Un composant démarre et arrête un traitement en tâche de fond comme il le souhaite

## ▶ Opérations

- ▶ **startService(...)**
- ▶ **stopService(...)**

## ▶ Mode *Bounded*

- ▶ Des composant (des "clients") établissent une connexion permanente afin d'interagir avec un service par le biais d'une interface

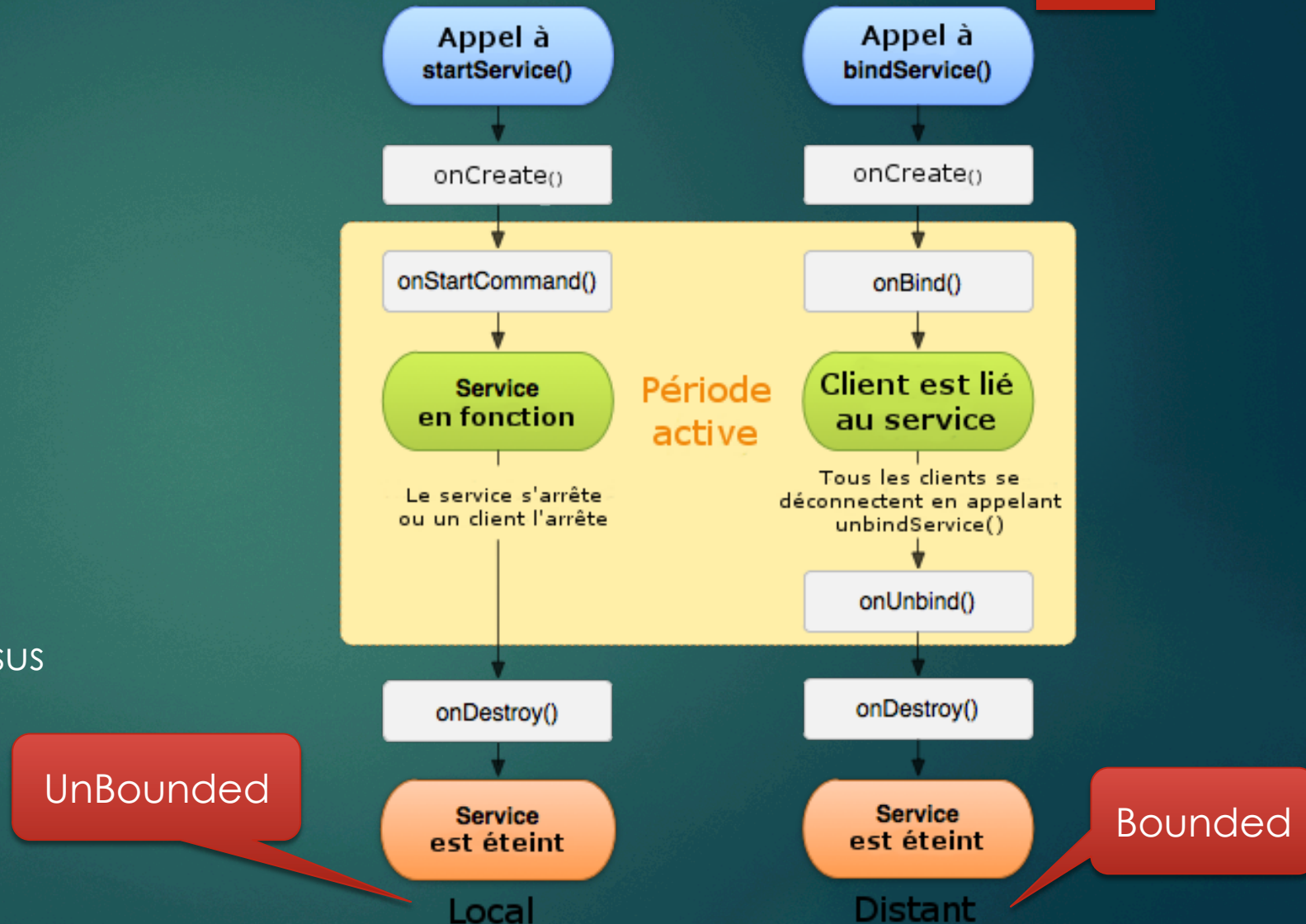
## ▶ Opérations

- ▶ **bindService(...)**
- ▶ **unbindService(...)**
- ▶ **plus** toutes les méthodes utiles que peut appeler l'activité sur le service



# Cycle de vie d'un service

- ▶ 5 méthodes de callback
  - ▶ onCreate()
  - ▶ onStart() SDK<2.0
  - ▶ onStartCommand()
  - ▶ onDestroy()
  - ▶ onBind()
  - ▶ onUnbind()
- ▶ S'exécute dans le processus courant





# Mode *Unbounded*

- ▶ Etape 1 : créer une Activity qui va contrôler le service
- ▶ Etape 2 : créer une classe qui étend `android.app.Service`.
  - ▶ Implémenter **`onCreate()`** : initialisation des ressources
  - ▶ Implémenter **`onStartCommand()`** : effectue le traitement et contient l'Intent envoyé
  - ▶ Implémenter **`onDestroy()`** : libère les ressources
- ▶ Etape 3 : déclaration dans le Manifest comme fils de l'élément `<application>`
  - ▶ **`<service android:name=".MyService"></service>`**
- ▶ Etape 4 : ajouter le code de contrôle du Service dans l'Activity
  - ▶ *// création du service et appel à `startCommand()` sur le service*  
**`startService(new Intent(MyActivity.this, MyService.class))`**
  - ▶ *// arrêt du service*  
**`stopService(new Intent(MyActivity.this, MyService.class))`**



# Code générique d'un Service *Unbounded*

```
import android.app.Service;

public class AccountChecking extends Service {

    @override
    public void onCreate() {
        //Allocation des ressources
    }

    @override
    public void onStartCommand(Intent intent, int flags, int startId) {
        //Code du service
    }

    @override
    public void onDestroy() {
        super.onDestroy();

        //Désallocation des ressources
    }
}
```



# Mode *Bounded*

- ▶ Etape 1 : créer une `Activity` qui va contrôler le service et qui va pouvoir appeler des méthodes sur le service. Le service durera le temps de l'activité qui l'a créé.
- ▶ Etape 2 : créer une classe qui étend `android.app.Service` :
  - ▶ Implémenter `onCreate()`
  - ▶ Implémenter `onDestroy()`
  - ▶ Implémenter `onBind()` : retourne la liaison
  - ▶ Implémenter `onUnbind()` : récupère la liaison
- ▶ Etape 3 : déclaration dans le Manifest comme fils de l'élément `<application>`
  - ▶ `<service android:name=".MyService"></service>`
- ▶ Etape 4 : ajouter le code de contrôle du Service dans l'Activity
  - ▶ Appel à `bindService(...)` : récupère la liaison sur le Service
  - ▶ Appel à `unbindService(...)` : libère la liaison sur le Service

La méthode `onStartCommand` ne sera pas appelée



# Mode *Bounded*

## ► MyService

1. Créer un **Binder** : le service crée un objet `mBinder` de type `Binder`, qui implémente la méthode `getService()` en retournant l'instance du service.
2. Implémenter **onBind()** : le service retourne `mBinder`
3. Implémenter les méthodes `public` supplémentaires que l'activité pourra appeler sur le service grâce à la liaison

## ► MyActivity

1. Créer un **ServiceConnection** qui implémente l'interface `ServiceConnection` (méthodes `onServiceConnected` et `onServiceDisconnected`) qui récupère dans l'activité l'objet de liaison
2. Appel à **bindService()** en lui passant le `ServiceConnection` en paramètre
3. Appel à **unbindService()** en lui passant le `ServiceConnection` en paramètre



# TP 05 : Tâches de fond – Les services

- ▶ A récupérer ici :
  - ▶ <http://www.labri.fr/~zemmari/pam>