LEZIONE 4 - 27/09/2017

https://github.com/giuseppegrosso/android.git

- Activity. Componenti che sovrintendono alla gestione della UI e controllano l'interazione con l'utente.
- Fragment. Componenti che consentono un'approccio modulare nella realizzazione e gestione di interfacce utente.
- Intent. Componenti che modellano in modo flessibile la comunicazione e le richieste di task fra componenti del sistema.
- BroadcastReceiver. Componenti che consentono la ricezione di notifiche riguardo ad eventi di sistema o stabiliti dallo sviluppatore
- **Service**
- ContentProvider

SERVICE

- Un service è un componente fondamentale nello sviluppo Android.
- Si tratta di un componente che lavora in background senza interazione utente diretta ed in modo disaccoppiato dagli altri componenti.
- Un service viene trattato dal sistema in maniera privilegiata ed in caso di app in background è meno soggetto a terminazione da parte del sistema stesso.
- Use casi tipici di un service sono i task:
  - lunghi e/o CPU intensive
  - periodici e indipendenti dal ciclo di vita della UI
- Esempi tipici sono il download di file, il polling di un servizio, ma anche la localizzazione, il geofencing ed la riproduzione di media.

SERVICE

- A livello di processo un Service condivide lo stesso Thread del resto dell'applicazione (sono detti Local Services)
- Per questo motivo è necessario prevedere operazioni asincrone se si tratta di task bloccanti o CPU intensive.
- La gestione dei thread associata al mondo dei service è uno dei task più critici nello sviluppo Android, per questo Google ha nel tempo fornito soluzioni semplificate agli sviluppatori (es. IntentService).
- Lo sviluppatore può decidere di avviare un Service in un processo autonomo sia per renderlo disponibile ad altre app sia per disaccoppiare dal main thread della app. Si complica notevolmente lo scambio di informazioni fra il Service ed il resto del mondo (IPC + AIDL).

SERVICE

- A livello di processo un Service condivide lo stesso Thread del resto dell'applicazione (sono detti Local Services)
- Per questo motivo è necessario prevedere operazioni asincrone se si tratta di task bloccanti o CPU intensive.
- La gestione dei thread associata al mondo dei service è uno dei task più critici nello sviluppo Android, per questo Google ha nel tempo fornito soluzioni semplificate agli sviluppatori (es. IntentService).
- Lo sviluppatore può decidere di avviare un Service in un processo autonomo sia per renderlo disponibile ad altre app sia per disaccoppiare dal main thread della app. Si complica notevolmente lo scambio di informazioni fra il Service ed il resto del mondo (IPC + AIDL).

SERVICE

- Il sistema operativo fornisce una serie di Service a cui lo sviluppatore, previa richiesta di permessi nel AndroidManifest.xml può accedere.

- Alcuni esempi:

  - Call (android.service.voice.*)

  - Media (android.service.media.*)

  - Message (android.service.textservice.*)

  - Notification (android.service.notification.*)

- Lo sviluppatore può effettuare il *binding* a questi servizi e accedere alle info necessarie se l'utente concede i permessi alla app.

SERVICE

- Chiaramente è possibile per lo sviluppatore definire Service custom.
- La definizione di un service passa attraverso due step. La dichiarazione nel file AndroidManifest.xml

```
<service
  android:name="AFService"
  android:label="@string/service_name"
  >
      ... filters etc come in una
          activity
</service>
```

- E' l'implementazione di una classe che estende o Service o una delle classi derivate messe a disposizione da Google.

SERVICE

- Prima di tutto vediamo come realizzare un service estendendo direttamente Service e non una delle sottoclassi già predispose.

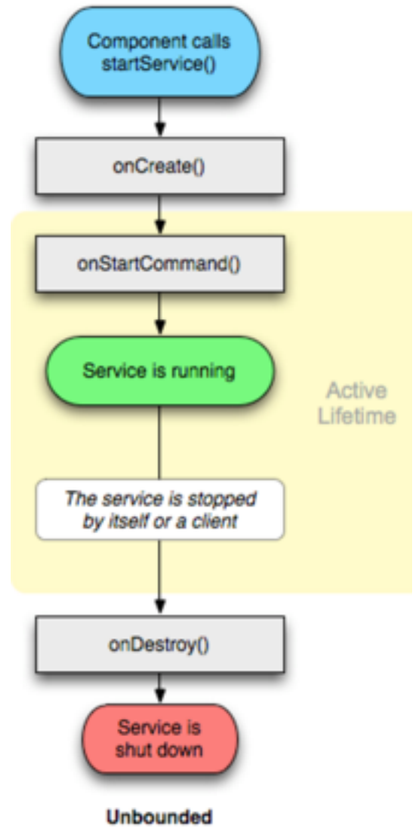
- Ci sono due metodi che è necessario implementare, corrispondenti alle due modalità con cui un service può essere avviato dagli altri componenti.

```
public class CleviriaService extends Service {

  @Override
  public int onStartCommand(Intent intent, int flags, int startId) {
    // Inizializzazione custom, ritorna un flag
    return Service.START_NOT_STICKY;
  }

  @Override
  public IBinder onBind(Intent intent) {
  // una istanza di IBinder è necessaria per comunicare con il Service in caso di bind()
    return null;
  }
}
```

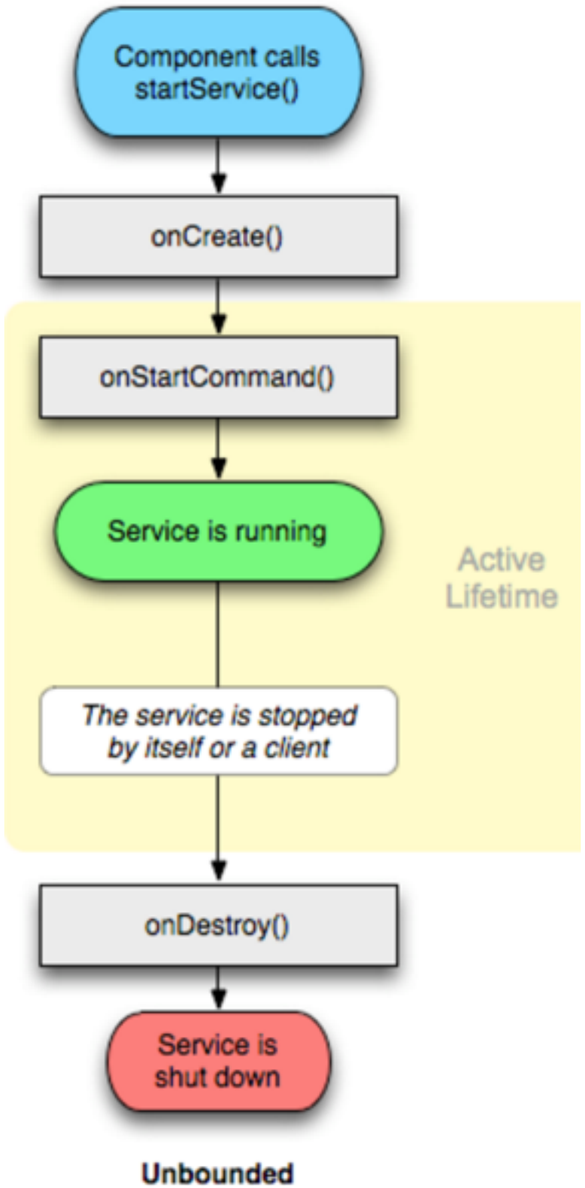
UNBOUNDED SERVICE

- Il primo metodo prevede di invocare context.startService() da un altro componente qualsiasi (Activity, Receiver, o classe in possesso di una istanza di Context)

```
// creazione nuovo Intent esplicito
Intent i= new Intent(context, MyService.class);
// opzionalmente si possono aggiungere dati come già visto
i.putExtra("KEY", "VALUE");
// avvio del servizio
context.startService(i);
```



UNBOUNDED SERVICE



- *onCreate()* viene chiamato una volta sola
- *onStartCommand()* viene chiamato ogni volta che si invoca *startService()* anche se il service è già in stato RUNNING
- Il service prosegue la sua attività ed usa risorse fino a che non viene stoppato in modo esplicito
  - invocando *stopSelf()* dall'interno del service stesso
  - invocando *context.stopService()* dall'esterno
- Il service prima di terminare invoca *onDestroy()*

SERVICE STARTED FLAG

```
@Override
public      int      onStartCommand(Intent      intent,      int      flags,      int      start
                        //    l'intero    ritornato      determina      il    comportamento
}
```

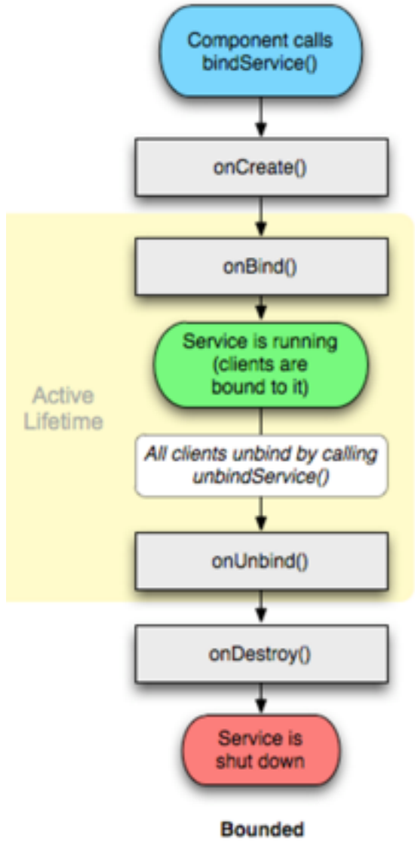
Valore	Comportamento
START_STICKY	Se il Service viene terminato dal sistema viene automaticamente riavviato, passando però un intent *null* all'*onStartCommand*.
START_NOT_STICKY	Se il Service viene terminato dal sistema non viene riavviato
START_REDELIVER_INTENT	Se il Service viene terminato dal sistema viene automaticamente riavviato, passando però l'intent originale all'*onStartCommand*.

- Default (ritornato da super.onStartCommand()) è START_STICKY

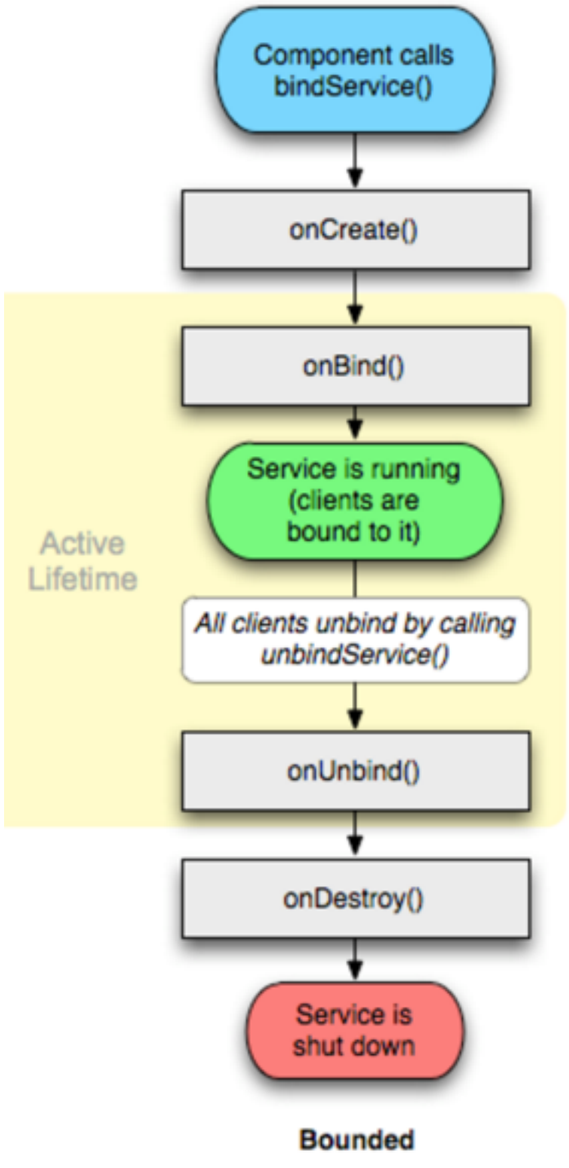
BOUNDED SERVICE

- Un service può anche essere detto *BOUNDED SERVICE* se viene avviato chiamando il metodo *bindService()* da un componente che estende Context o è in possesso di una istanza di Context

```
Intent intent= new Intent(this, LocalWordService.class);

bindService(intent, mConnection,Context.BIND_AUTO_CREATE);
```



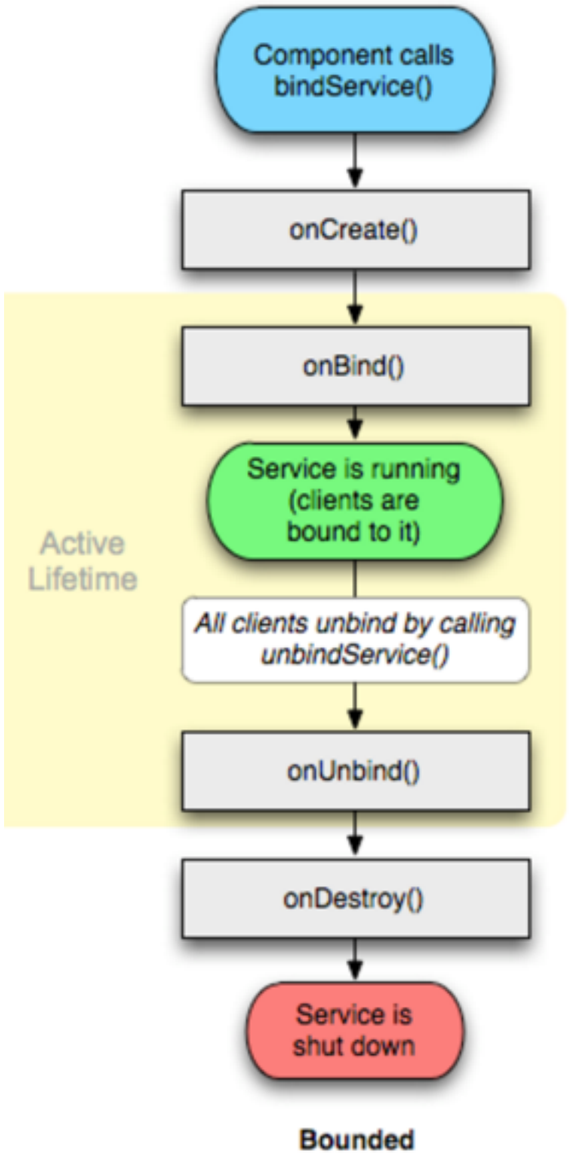
BOUNDED SERVICE



- Il binding richiede un'istanza di ServiceConnection che il "chiamante" passa a *bindService()*

```
CleviriaService s = null;

....

private ServiceConnection mConnection = new ServiceConnection() {

    public void onServiceConnected(ComponentName className, IBinder binder)
    {
      CleviriaService.MyBinder b = (CleviriaService.MyBinder) binder;
      s = b.getService();
      Log.d("CL","Connesso a CleviriaService");
    }

    public void onServiceDisconnected(ComponentName className) {
      s = null;
    }
};

  ...

  Intent intent= new Intent(this, CleviriaService.class);
  bindService(intent, mConnection,Context.BIND_AUTO_CREATE);
```

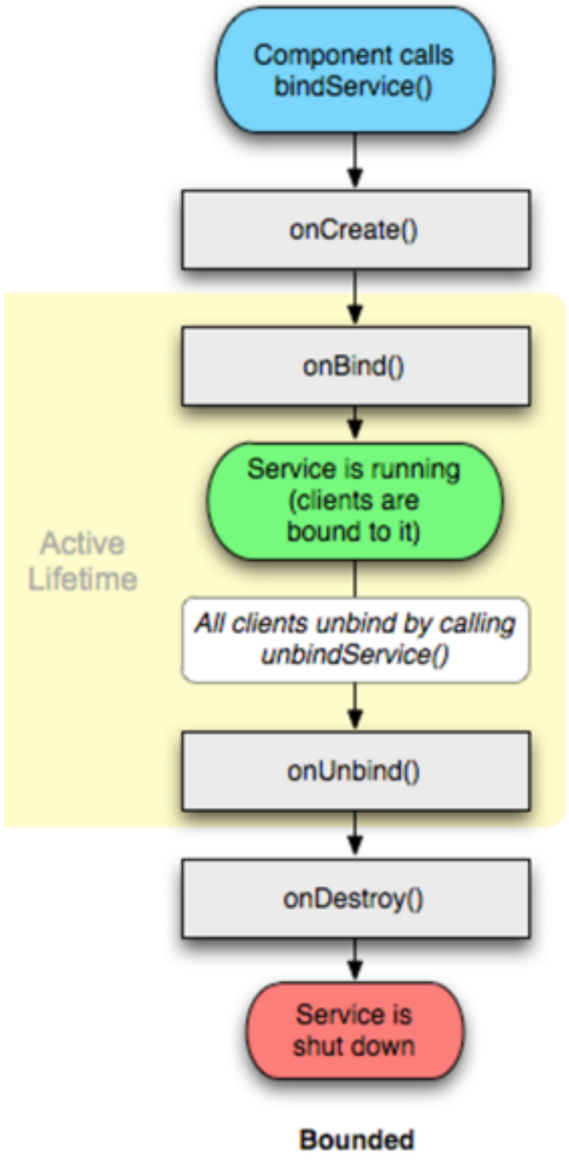
BOUNDED SERVICE



- Lato service è necessario implementare *onBind()* e ritornare un'istanza valida di IBinder che fornisca al chiamante un accesso al service stesso

```
private final IBinder mBinder = new MyBinder();

  @Override
  public IBinder onBind(Intent arg0) {
    return mBinder;
  }

  public class MyBinder extends Binder {
    CleviriaService getService() {
      return CleviriaService.this;
    }
  }
```

BOUNDED SERVICE



- Un BOUNDED SERVICE è un po' come un server in un paradigma client/server. Viene avviato per fornire un servizio a dei client che possono interagire con il server attraverso un riferimento ad esso che ottengono durante il *binding*.
- La disconnessione si attua chiamando *unbindService()* e passando la stessa istanza di ServiceConnection
- A disconnessione completata viene invocato *onServiceDisconnected*
- *Un BOUNDED SERVICE vive fino a quando ha almeno un client in stato BOUND*

INTENT SERVICE

- Un Intent Service ha un lifecylce simile ad un service unbounded. La differenza é la classe base: IntentService, e non Service.
- Pensato per *one-time-task* (tipo download/upload risorsa generica)
- Si avvia allo stesso modo

```
Intent intent = new Intent(this, WGetService.class);
// add infos for the service which file to download and where to store
intent.putExtra(WGetService.FILENAME, "luca.png");
intent.putExtra(WGetService.URL,
        "http://elbuild.it/images/luca.png");
startService(intent);
```

- Ha una callback aggiuntiva *onHandleIntent()* che facilita di molto il lavoro dello sviluppatore.

INTENT SERVICE

- Fra *onStartCommand* e *onHandleIntent* il sistema operativo si preoccupa di creare un *Handler* che gestisce il lavoro del service su un Thread separato.
- Nessuna necessità di preoccuparsi di sincronizzazione
- Il service termina autonomamente al termine del lavoro, ovvero quando l'esecuzione del metodo *onHandleIntent* è terminata.
- Non esiste policy di restart, nè stato all'interno del *Service*. Una volta che è terminato se ne può solo lanciare uno nuovo.
- In funzione dell'hardware si possono lanciare anche N istanze di IntentService in parallelo (scelta non comune).

INTENT SERVICE

- Una prima forma di comunicazione sono gli Intent passati a startService() e bindService(). MONODIREZIONALE e non ripetibile.
- Una forma di comunicazione MONODIREZIONALE nel senso opposto sono sempre gli Intent, inviati sotto forma di sendBroadcast() verso un receiver.
- Nel caso di Bounded Service l'istanza ottenuta tramite il binding può essere utilizzata per invocare in modo sincrono metodi pubblici del service da parte dell'activity.
- Handler e ResultReceiver sono classi che consentono una comunicazione bidirezionale con un service. Abbastanza complesso.
- Android Interface Definition Language, in caso di service esposti pubblicamente.
- 3rd party libraries. Otto Bus o Event Bus. Implementano Event Bus design pattern.

INTENT SERVICE EXAMPLE

Q&A
Question and answer

https://github.com/giuseppegrosso/android.git