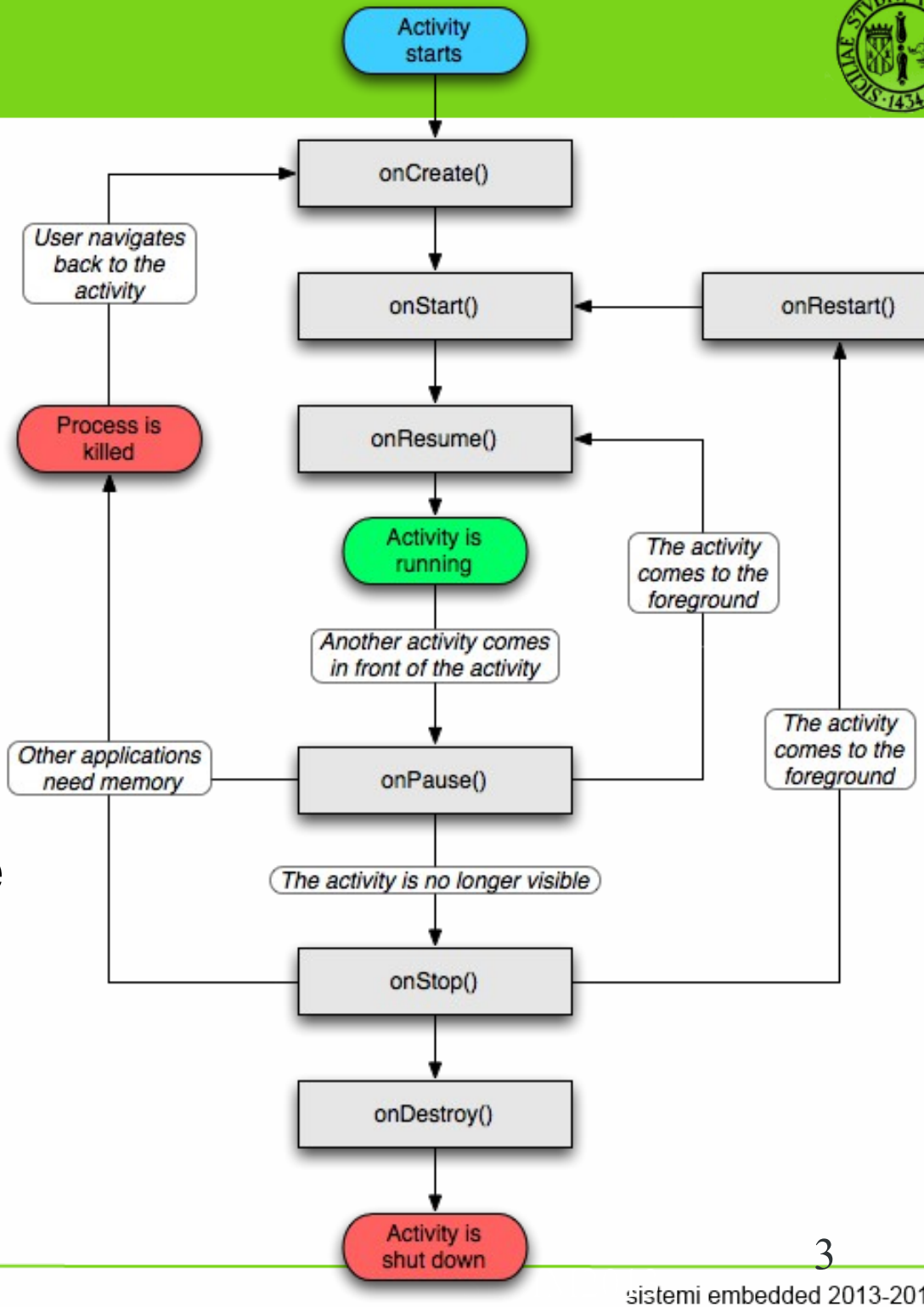
Ciclo di vita di Activity, Service e Processi

Introduzione

In questa lezione analizzeremo il ciclo di vita delle Activity e dei Service mettendo in evidenza i criteri utilizzati da Android nella gestione dei relativi processi

Activity 1/3

- Pur essendo possibile avviare più applicazioni contemporaneamente soltanto una può occupare il display, mentre le altre saranno ”nascoste” in background
- Questo è il motivo per cui, il concetto di chiusura è secondario e normalmente non troveremo il punsante ”Esci”



Activity 2/3

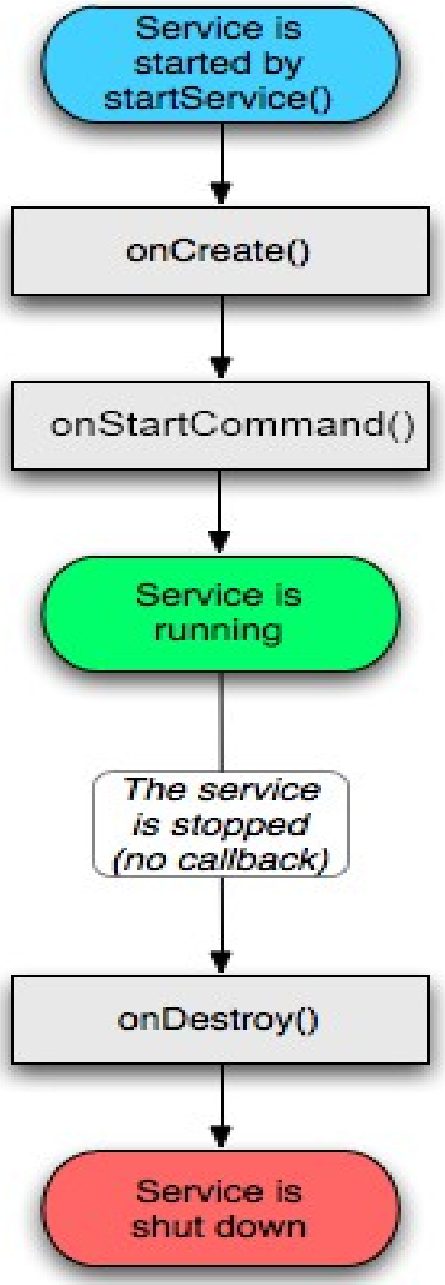
- **OnCreate(Bundle)**: e' invocato quando l'activity viene avviata per la prima volta. Il Bundle savedInstanceState serve per riportare l'Activity nello stesso stato in cui si trovava la precedente istanza dell'Activity terminata.
- **OnStart()**: è invocato quando l'activity sta per essere visualizzata
- **onResume()**: è invocato non appena l'activity inizia ad "interagire" con l'utente
- **onPause()**: è invocato non appena l'activity sta per essere ”ibernata” (per es. e' stata avviata un'altra activity)
- **onStop()**: è invocato nel momento in cui l'activity non e' piu' visibile all'utente.

Activity 3/3

- **onRestart()**: è invocato quando l'Activity sta per essere riavviata dopo essere stata precedentemente arrestata
- **onDestroy()**: è invocata poco prima che l'Activity sia distrutta
- **onSaveInstanceState(Bundle)**: è invocata per salvare lo stato dell'Activity
- **onRestoreInstanceState(Bundle)**: è invocata solo se in precedenza è stato salvato uno stato

Ciclo di vita di un Service

- Quando viene invocato il metodo startService() il sistema verifica se tale servizio è in escuzione altrimenti esegue onCreate()
- Successivamente è invocato il metodo onStartCommand() e il servizio è finalmente nello stato di esecuzione dove rimane fino a quando non sarà richiamato stopService() oppure stopSelf().



Service 1/2

- **onCreate()**: a differenza delle Activity non presenta alcun parametro. Creato il servizio viene invocato il metodo onStartCommand(). Viene invocato una sola volta.
- **onStartCommand()**: invocato questo metodo il servizio resterà in esecuzione e rimarrà in questo stato fino a quando non sarà invocato stopService() da parte dell'Activity che lo ha generato, oppure stopSelf() da parte del servizio stesso. Può essere invocato più volte in seguito all'esecuzione di onStartService() da parte dell'Activity.

Service 2/2

- **onDestroy()**: l'invocazione di stopService() da parte dell'Activity produrrà la chiamata di questo metodo del Servizio. In seguito a questa chiamata il servizio sarà eliminato

Applicazioni e Processi

- Per ogni applicazione che viene eseguita, Android avvia un nuovo processo Linux
- Di default, tutti i componenti di una data applicazione (per es. Activity + Service) sono eseguiti all'interno dello stesso processo
- Se al momento di avviare una applicazione esiste già un processo assegnato ad essa perchè un suo componente è stato eseguito in precedenza, allora tale applicazione sarà eseguita all'interno di questo processo

Applicazioni e Processi

- Nel caso in cui la memoria a disposizione per eseguire una applicazione in *foreground* (visibile all'utente) ovvero per eseguire un importante processo non sia disponibile, allora il sistema può decidere di eliminare uno o più processi in *background* secondo una logica ben definita
- Ad ogni processo viene assegnata una sorta di importanza gerarchica in base ai suoi componenti in esecuzione ed al loro stato
- Ovviamente i processi di minore importanza saranno i primi ad essere eliminati per liberare memoria

Applicazioni e Processi

Sono state definite 5 tipologie di processi ordinate per importanza decrescente:

- Foreground Process
- Visible Process
- Service Process
- Background Process
- Empty Process

Foreground Process

I processi Foreground sono quelli che eseguono componenti di interazione con l'utente:

- Activity in cima allo Stack
- Azioni di un BroadCastReceiver (OnReceive())
- Metodi di callback nella gestione di un servizio come Start(), Create() o Destroy()

Normalmente il numero di questo tipo di processi è limitato. Android potrà decidere di eliminarne uno come ultima chance per risolvere il problema di memoria

Visible Process

Sono processi che pur non essendo direttamente visibili all'utente (non possiedono componenti in Foreground, si trovano in uno stato in cui possono comunque avere effetto su quello che l'utente vede visualizzato.

- Activity non in Foreground che esegue OnPause()
- Service connesso ad un'Activity in Foreground.

Service Process

- Sono processi che eseguono Service
- Pur non interagendo direttamente con l'utente, Android li ritiene più importanti delle Activity in Background
- L'esempio classico è la riproduzione di un file audio: pur non interagendo direttamente con l'utente, questo accetterà malvolentieri l'interruzione dell'ascolto del brano. Pertanto Android proverà ad eliminare uno o più processi meno importanti prima di eliminare un Service Process

Background Process

- Sono processi che si occupano di eseguire Activity non più in Foreground per i quali è stato invocato il metodo OnStop()
- Il loro numero è in genere superiore a quello delle altre categorie
- L'eliminazione all'interno di questa categoria tiene in considerazione il criterio di Least Recent Used (LRU)
- Per questi processi è comunque definito un meccanismo di salvataggio e ripristino

Empty Process

- Sono i processi candidati per l'eliminazione.
- Non eseguono alcun componente
- Il motivo per cui non vengono eliminati nel momento in cui sono svuotati dai componenti che eseguivano è legato a strategie di caching
- Gli empty Process più vecchi saranno velocemente eliminati, mentre per quelli più recenti è più probabile che l'utente richiami un suo componente precedentemente visualizzato