Tools of the trade

<https://www.youtube.com/watch?time_continue=2&v=AeoeD7K8vKI>

Cosas chulas para desarrollar major

<http://www.technotalkative.com/lazy-android-part-7-useful-tools/>

otra charla de uso de la batery:

https://www.youtube.com/watch?v=LO\_Swql0zVg

**Memory Leak**

<https://www.youtube.com/watch?v=mU1VcKx8Wzw>

Proyecto android de la banda de square(retrofit, Picasso, dagger) qu ese usa para detectar los memoryleaks

<https://github.com/square/leakcanary>

cosas interesantes:

https://developer.android.com/intl/es/tools/debugging/debugging-memory.html

1 ¿Qué es un Memory Leak?

2 ¿Cómo se produce o xq se produce un Memory Leak, (goteo de memoria)

0 la memoria (heap, stack)

* 1. garbage collector

Static variables not enter in collected Garbage.

Ejemplo del botón static, este tiene una referencia al activity y el activity una del window la cual tiene una referencia a toooodo lo pintado en pantalla.

Para el garbage collector estas variabels staticas se denominan GC Route, y el punto es que

El gc recicla o colecta toooodo salvo los GC Route, xq entiende que deben seguir en memoria, bien hecho el problema viene cuando su contexto fue eliminado y no se seteo explícitamente el static a null.

3 Herramientas

HAHA

Es un generador de código extra para poder analizar que está pasando con la memoria.

Crea un grafo uqe te permite ir haciaatras y ver quien es el responsble final de ese goteo de memoria

Memory Analizer

Lo que hace es un volcado a disco de toda la memoria(Heap) y te permite usar un pseudo SQL para analizar los datos.

Por ejemplo:

**SELECT** **\*** **FROM** INSTANCEOF android.app.Activity a **WHERE** a.mDestroyed **=** **true**

Esta query nos devolveria todas las activities que se han quedado en memoria como destruidas, pero que no han sido reaprovechadas por el GC.

Como ya vimos el GC se folla todo menos las GC Route, con lo cual una vez que hemos obtenido todas las activities que tienen goteos, vamos a buscar el origne del problema, estos GCRoutes. Para ello ejecutamos un método merge\_shortest\_paths mediante estos path vamos a ir viendo quien es el responsable final.

Como esto es un trabajo de chinos (MONO ABURRIDO) dios bajo y creo(leak canary)

LeakCanary

Sirve para detectar goteos de memoria mientras desarrollamos.

Los goteos de memoria son inevitables, ya que el propio API de Android los contiene y x tanto,no hay forma de evitarlos todos.

Pero con Leak Canary es posible que detectemos todos ellos y los solucionemos. El problema real detrás de un memory leak es que por su culpa, produzcamos un out of memory error.

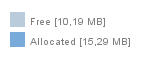
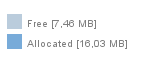
Donde se abre

Que Muestra

Como encontrar el responsable del leak

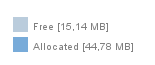
Demo practica:

Sin static ni gato, navegando de 1 a 2 pre GC init y post, tras back de 2 a 1 se queda igual 10,19



Sin static con gato, navegando de 1 a 2 pre GC init y post, tras back de 2 a 1 se queda igual 

Con static con gato, navegando de 1 a 2 pre GC init y post



Buenas practicas:

-Usar variables estáticas con conocimiento.

-Si usas campos staticos, ten muy en mente setearlos a null cuando destruyas el entorno en elque trabaja ese static.

Por ejemplo si usar un campo static en un activity, en el onDestroy del activity, recuerda setearlo a null, ya que sí no lo haces provocarás un goteo de memoria, manteniendo en memoria todos los padres(context) de ese Static, lo cual puede hacerse bola y terminar en un out of memory



­