Apuntes

Descargar estos apuntes

Tema 10. Corrutinas y Servicios

Índice

Corrutinas Kotlin

Una corrutina de Koytlin es un conjunto de sentencias que realizan una tarea específica, con la capacidad suspender o resumir su ejecución sin bloquear un hilo. Esto permite que tengas diferentes corrutinas cooperando entre ellas y no significa que exista un hilo por cada corrutina, al contrario, puedes ejecutar varias en un solo.

Las corrutinas hacen parte del paquete kotlinx.coroutines, por lo que necesitas especificar la dependencia correspondiente en en `build.gradle`. A día de hoy:

```
implementation "org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.1"
```

Las corrutinas en el desarrollo de Android, aunque son un elemento que va a permitirnos crear códigos más sencillos, al principio pueden parecer complejas debido a su nivel de abstracción.

Alcance de las corrutinas

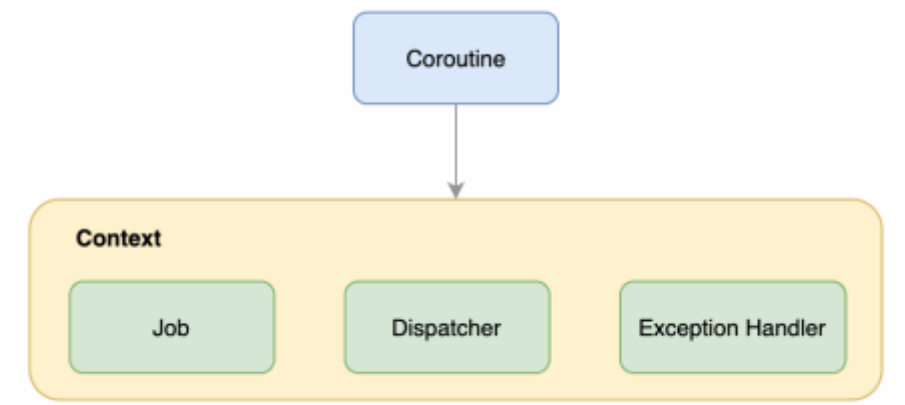
Cuando creamos coroutines asumimos de manera implícita dos aspectos muy importantes: El ámbito (CoroutineScope y GlobalScope) y el contexto (CoroutineContext).

El uso de `GlobalScope` permite crea corrutinas de nivel superior, esto quiere decir que tienen la capacidad de vivir hasta que termine la aplicación y es trabajo del desarrollador en control para su cancelación, por lo que no se aconseja usar este ámbito.

Para reducir este alcance, Kotlin nos permite crear los espacios donde queremos que se dé la concurrencia. Esto lo haremos mediante `CoroutineScope`. Cada vez que usamos un constructor de coroutines en realidad estamos haciendo una llamada a una función que recibe como primer parámetro un objeto de tipo `CoroutineContext`.

CoroutineContext

Las coroutines siempre se ejecutan en algún contexto que está representado por un valor del tipo CoroutineContext.



El contexto de Coroutine es un conjunto de varios elementos. Los elementos principales son el
Job de la Coroutine, su dispatcher y también su Exception handler.

- **Job** De acuerdo con la documentación oficial "Un Job es una cosa cancelable con un ciclo de vida que culmina con su finalización. Los trabajos de coroutine se crean con launch coroutine builder. Ejecuta un bloque de código especificado y se completa a la finalización de este bloque".
  La ejecución de un trabajo no produce un valor de resultado. Deberíamos utilizar una interfaz Deferred para un trabajo que produzca un resultado. Un trabajo con resultado (Deferred), se crea con el constructor async Coroutine y el resultado se puede recuperar con el método await(), que lanza una excepción si el Deferred ha fallado.
- **Dispacher** En Kotlin, todas las Coroutines deben ejecutarse en un dispatcher incluso cuando se ejecutan en el hilo principal. Las coroutines pueden suspenderse a sí mismas, y el dispatcher es el que sabe cómo reanudarlas.
  Para especificar dónde deben ejecutarse las coroutines, Kotlin proporciona tres Dispatchers que puedes utilizar:
  - `Dispatchers.Main`. Hilo principal en Android, interactúa con la UI y realiza trabajos ligeros como: llamar a funciones de suspensión, llamar a funciones de la interfaz de usuario y actualizar LiveData
  - `Dispatchers.IO`. Optimizado para la E/S de disco y red fuera del hilo principal: solicitudes de bases de datos, lectura/escritura de archivos; conexión en red.
  - `Dispatchers.Default`. Optimizado para el trabajo intensivo de la CPU fuera del hilo principal: ordenar una lista, análisis de JSON, utilidades

Dispatcher	Description	Uses
Dispatchers.Main	Main thread on Android	- Calling suspend functions - Call UI functions - Update LiveData
Dispatchers.IO	Disk and network IO*	- Database - File IO - Networking
Dispatchers.Default	CPU intensive work	- Sorting a list / other algorithms - Parsing JSON - DiffUtils

- **Exception Handler**. Este elemento es opcional del CoroutineContext, y nos permite manejar excepciones no capturadas. Y de esta forma mejorar la expericencia de usuario

Funciones de Suspensión

Las corrutinas se basan en las **funciones de suspension** que son funciones normales a las que se les agrega el modificador `suspend`, de esta manera se agregan dos nuevas operaciones (además de invocar / llamar y regresar):

- suspender - pausa la ejecución de la corrutina actual, guardando todas las variables locales. El hilo actual puede continuar con su trabajo, mientras que el código de suspensión se ejecuta en otro hilo.
- reanudar : continúa una corrutina suspendida desde el lugar donde se pausó cuando el resultado está listo.

Solo se pueden llamar a las funciones de suspensión desde otras funciones de suspensión, o utilizando un constructor de corrutinas para iniciar una nueva corrutina. Estas funciones pueden ejecutar una operación de larga duración y esperar a que se complete sin bloquearse. Hay que tener en cuenta que las funciones de suspensión se sincronizan automáticamente con otras variables y funciones del hilo principal.

Vamos a poner en funcionamiento todos estos conocimientos retomando el ejercicio resuelto del bloque 6 **ProgresIndicators**, mejorando la funcionalidad.