Coroutine 讀書會 S1E6
補充資料

關注第一梯次 Corotine讀書會活動:
https://tw.kotlin.tips/study-jams/5

共讀書本及討論群

- 導讀投影片:
  https://drive.google.com/file/d/1qSZA7R3vexy7PbtWV6I8fr2mH8EmGepS/view?usp=sharing
- 活動錄影:https://youtu.be/UB5f0TKwW6E
- 共讀書本:https://www.tenlong.com.tw/products/9787111655916
- 讀書會討論群
  - Line 群 https://line.me/R/ti/g/jAG_c1DMaO (請先加聖佑 line id:shengyoufan 後傳訊息給聖佑, 聖佑再邀請入群)
  - Telegram 群 https://t.me/joinchat/DAUjOBrfu7_05sa9YYb5FQ

導讀筆記

- Kotlin official library
  - CoroutineLite
  - 主要的構成:
    i. Core: Channel, Flow
    ii. Ui: 包括 android, javafx, swing
    iii. reactive 相關: reactive, reactor, rx2
    iv. Integration: jdk8, guava, slf4j…等等等
- Coroutine Runs - launch(), 會回傳一個Job實體
- Coroutine start option:Default, Atomic, Lazy, Undispatched.

	立刻調度	保證執行	備註
Default	O	X	可能在執行前就被取消
Atomic	O	O	原子化(調度/執行同時)
Lazy		X	只有在被呼叫時才會調度, 但如果在此之前就被取消, 則會進入異常結束狀態
Undispatched		O	立即執行

- Coroutine dispatcher

    - Dispatchers.Default
    - Dispatchers.IO
    - Dispatchers.Unconfined
    - asCoroutineDispatcher：是個能將Java Executor轉換為Kotlin Dispather的extention function，需要自行將Dispatcher關閉
- Exception handler
    - 預設handler例外處理優先順序
        i. 若為CancellationException則忽略
        ii. 若當前context中有Job，則將該Job取消
        iii. 透過CoroutineExceptionHandler呼叫Thread.uncaughtExceptionHandler
- Cancellation
    - Coroutines在實際執行cancel之前，需要先確認當前的CoroutineScope.isActive，如果值為false則不會執行cancel
- Timeout
    - withTimeOut：當逾時會丟出Exception
    - withTimeOutOrNull：當逾時會回傳null

- Asynchronous data stream
- Channel
    - similar to Java's java.util.concurrent.BlockingQueue
    - 實際上會建立兩個Coroutines，一個用來產生值，一個用來接收
- Buffering
    - 在建立Channel時，可以同時設定此Channel容量

	Buffer size	Suspend situation
RENDEZVOUS	0	send掛起直到receive執行 receive掛起直到send執行
UNLIMITED	unlimited	send永遠不會被掛起
CONFLATED	1	send永遠不會被掛起，但receive呼叫前回傳的值會被忽略
others	根據設定容量	當buffer被占滿時send掛起，當buffer清空時receive掛起

- Close Channel
    - channel.close()，表示不會再有新的值需要產生
    - 當呼叫close()，closedForSend會先變為true，channel不會再產生新的回傳值，而closedForReceive一直到所有producer產生的值被接收完才會變為true
- BroadcastChannel
    - Channel：一對一
    - BroadcastChannel：可以一對多

- Flow
    - Cold streams, declarative, reactive, asynchronous yet sequential
    - Flow builder中的code只有在collect被呼叫時才會執行
    - Flow Cancellation：when flow is suspend in a cancellable suspending function
    - buffer()：with buffering 比 Without Buffering 效率高
    - Conflation：conflate() 透過 dropping emitted values 提高效率
    - Composing Flows：zip, Flatten
    - Operators
- Avoiding a shared mutable state
    - 使用Mutex, 類似於blocking code中的asynchronized
    - Pure function
    - Functional programming

參考資料

- Atomic 與 Mutex 的效能比較
  https://magiclen.org/rust-atomic-mutex-performance-measurement/
-