Kotlin: Coroutines and More

Contact Info

Ken Kousen

Kousen IT, Inc.

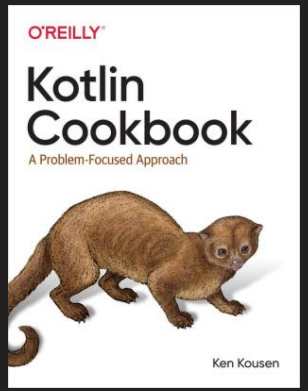
ken.kousen@kousenit.com

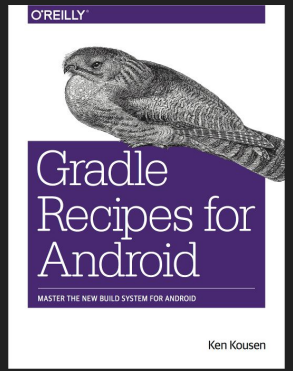
http://www.kousenit.com

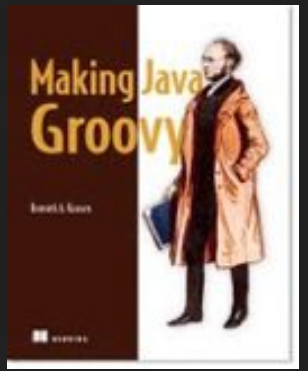
http://kousenit.org (blog)

@kenkousen (twitter)

https://kenkousen.substack.com (newsletter)







Certified Training Partner

GitHub repository

All code examples in the Kotlin Cookbook repo

https://github.com/kousen/kotlin-cookbook

Kotlin

JetBrains created and maintains the language

Provides null safety at the compiler level

Statically typed and statically bound by default

Runs on the JVM → Clean interoperability with Java

Kotlin

- Home page is https://kotlinlang.org
- Many code simplifications borrowed from other languages
  - Closures similar to Groovy
  - Typing similar to Scala
  - Coroutines similar to .Net (and others)

URL extension functions

```
fun URL.readBytes(): ByteArray
```

Read content of URL as a byte array

```
fun URL.readText(charset: Charset = Charsets.UTF_8): String
```

Read content of URL as a string with supplied encoding

Astronauts in Space

Access REST service and process JSON data

astro.AstroRequest.kt,
astro.AstroRequestTest.kt

Sequences

Methods like "map", "filter" are added to collections

The "`asSequence()`" method converts collection to sequence

- Like Java streams
- Evaluated element at a time
- No data processed unless there is a terminal expression

Pair

Good example of tuple with extension function

Used to create maps

See "to" function

collections.MapTests.kt

Measuring time

```
inline fun measureTimeMillis(block: () -> Unit): Long

inline fun measureTimeMicros(block: () -> Unit): Long

inline fun measureNanoTime(block: () -> Unit): Long
```

misc.MeasureTime.kt

Scope functions: apply

Context object is "this", return value is the object itself

Use for configuration → "apply the following changes to the object"

From Ktor server:

```
val server = embeddedServer(Netty, port = 8080,
    module = Application::mymodule
).apply {
    start(wait = false)
}
```

Scope functions: let

Context arg is "it", return is the lambda result

"let's do the following to the object"

```
fun processNullableString(str: String?) =
    str?.let {
        when {
            it.isEmpty() -> "Empty"
            it.isBlank() -> "Blank"
            else -> it.capitalize()
        }
    } ?: "Null"
```

let with Elvis

<nullable>?.let { … } ?: <default>

scope.let_demo.kt
scope.ProcessStringTest.kt

Scope functions: also

Context is argument, return value is the object itself

"and also do the following"

Great for side-effects, like printing

```
@Test
fun `find all primes less than 20`() {
    primesLessThan(20)
        .also(::println)
}
```

Sorting

```
fun <T : Comparable<T>> Array<out T>.sort()
```

applies to arrays, collections, …

```
inline fun <T, R : Comparable<R>> Iterable<T>.sortedBy(
    crossinline selector: (T) -> R?): List<T>
```

```
fun <T> MutableList<T>.sortWith(comparator: Comparator<in T>)
```

collections.comparisons.kt

Delegates: lazy

Delay computation until needed

```
val ultimateAnswer: Int by lazy {
    println("computing the answer")
    42
}
```

Delegates: observable

Execute lambda when value changes

```
var watched: Int by Delegates.observable(1) {
    prop, old, new ->
        println("${prop.name} changed from $old to $new")
}
```

Delegates: vetoable

Only change a property value if permitted

```
var checked: Int by Delegates.vetoable(0) {
    prop, old, new ->
        println("Change ${prop.name} from $old to $new")
        new >= 0
}
```

Delegates: property delegates

Delegate property values to interfaces

delegates.phones.kt
delegates.SmartPhoneTest

TODO function

kotlin.TODO

```
public inline fun TODO(): Nothing =
    throw NotImplementedError()
```

KotlinVersion

kotlin.KotlinVersion class → determine current Kotlin version

`KotlinVersion.CURRENT` → 1.3.71 (major.minor.patch)

Implements `Comparable`, so can use less than / greater than

KotlinVersion

```
override fun equals(other: Any?): Boolean {
    if (this === other) return true
    val otherVersion = (other as? KotlinVersion) ?: return false
    return this.version == otherVersion.version
}
```

- `shl` to create `int` version from major/minor/patch
- implements `Comparable`
- constants in companion object
- great example of `equals` implementation

Much ado about Nothing

kotlin.Nothing

```
public class Nothing private constructor()
```

That's the entire class

- Return type when function only throws exception
- Nothing? when var/val assigned to null without type info
- Nothing is a subtype of every other type
    - Wait, what?

lateinit

lateinit modifier showing not yet ready, but will be

oop.lateinitdemo,
oop.LateInitDemoTest

File I/O

useLines function

sorting and grouping

io.FileIO.kt,
io.FileIOTest

Tail Recursion

tailrec keyword

functional.algorithms.kt

fold, reduce

fold takes initial argument (identity) and lambda

reduce just takes lambda

functional.algorithms.kt,
functional.AlgorithmsTest.kt

Coroutines

- Confusing, but simpler than the alternatives
- Too many classes, but small number of categories
- Too many use cases, but only a few occur in practice

Coroutines

So what to you need to know?

Scopes, Builders, Dispatchers

- Context elements like Job and SupervisorJob
- CoroutineExceptionHandler

Scope

- Start with a scope (except when you don't, but go with it here)
  - All coroutines must run in a scope
  - A CoroutineScope manages one or more related coroutines
  - The heart of structured concurrency

Scope

On Android, the KTX extension libraries provide:

- lifecycleScope
- viewModelScope
- Both are aware of component life cycles
  - Cancel automatically when component is cleared

Scope

- Also available is coroutineScope builder
  - Creates a coroutine scope
  - Does not complete until all launched children complete
  - Is a suspend function
    - Does not block a thread, unlike runBlocking

coroutineScope function

```
1 suspend fun <R> coroutineScope(
2     block: suspend CoroutineScope.() -> R
3 ): R (source)
```

coroutineScope

coroutineScope creates an instance of the CoroutineScope class

Invokes the specified suspend function block with this scope

Builders

Use a builder to launch a coroutine

launch

async

launch

The launch function creates a new coroutine

Returns a Job, but not a result

- Does not block the current thread
- Coroutine is cancelled when the resulting job is cancelled

launch Builder function

```
fun CoroutineScope.launch(
    context: CoroutineContext = EmptyCoroutineContext,
    start: CoroutineStart = CoroutineStart.DEFAULT,
    block: suspend CoroutineScope.() -> Unit
): Job
```

async

async returns a Deferred

A Deferred is a Job with a result

Running coroutine cancelled when resulting deferred is cancelled

Execute by calling await()

Dispatchers

Dispatchers determine which thread or threads a coroutine uses

This is part of the coroutine context

In Android, need to be off the main thread for networking or db access

- Return to the main thread to update UI

launch and async accept an optional CoroutineContext parameter

- Use that to specify which dispatcher to use

Dispatchers

Available Dispatchers:

Main → used for updating UI elements

Default → used for long-running processes

shared pool of threads

IO → used for offloading blocking IO tasks, like networking

shared pool of threads

Dispatchers

Use withContext function to switch from one dispatcher to another

Calls a suspend block with a given context

Suspends until it completes

Returns a result

withContext function

```
1 suspend fun <T> withContext(
2     context: CoroutineContext,
3     block: suspend CoroutineScope.() -> T
4 ): T
```

Job and SupervisorJob

Jobs form a hierarchy of parents and children

Can instantiate a Job, but builders also create them

When a parent coroutine is cancelled, all its children are cancelled

Children of a SupervisorJob can fail independently of each other

Samples

In GitHub repo:

coroutines.coroutines_N.kt where N == 1..5

coroutines.openweathermap.kt

Coroutine Docs

Reference docs:

https://kotlin.github.io/kotlinx.coroutines/

Coroutines by example:

https://github.com/Kotlin/kotlinx.coroutines/blob/master/coroutines-guide.md

(Also part of basic User Manual)

Excellent Udemy Course

Excellent Udemy (video) course on coroutines on Android:

Mastering Kotlin Coroutines for Android Development

https://www.lukaslechner.com/coroutines-on-android/

Lukas Lechner

GitHub repository (free, of course):

https://github.com/LukasLechnerDev/Kotlin-Coroutine-Use-Cases-on-Android

GraalVM: native image

```
$ sdk install java 20.2.0.r11-grl
$ sdk use java 20.0.2.r11-grl
$ gu install native-image

$ kotlinc-jvm antarctica.kt
$ kotlin AntarcticaKt
$ kotlinc-jvm antarctica.kt -include-runtime -d antarctica.jar
$ native-image -jar antarctica.jar
```

GraalVM: native image

$ time kotlin AntarcticaKt → 0.173

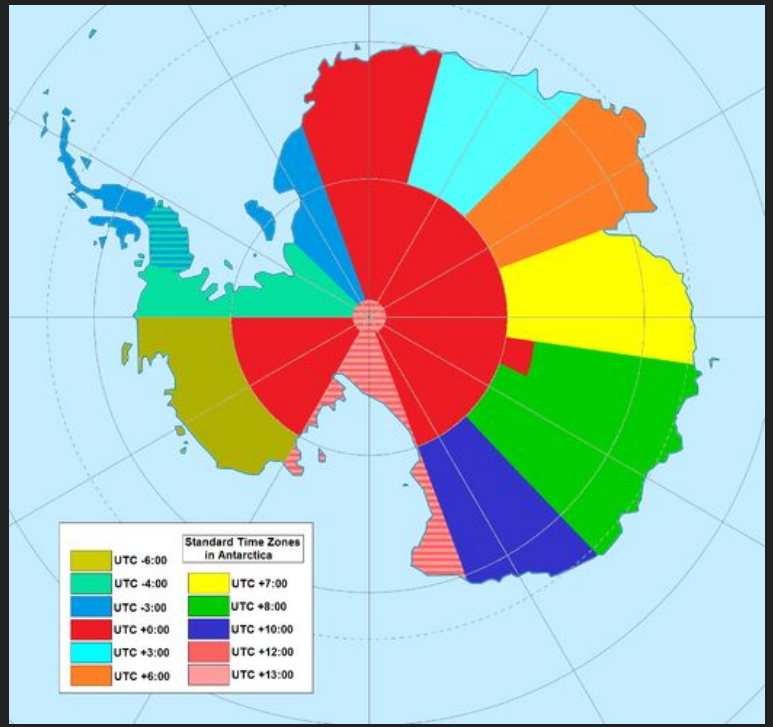
$ time java -jar antarctica.jar → 0.138

$ time ./antarctica → 0.009

https://www.graalvm.org/docs/reference-manual/native-image/

Gradle plugin: com.palantir.graal

Antarctica Time Zones



Conclusions

- Like Java, but more flexible
- Null safety
- Lots of nice features with lambdas
- Growing library
- Data classes are sweet
- Coroutines are a mess, but doable and getting better
- Please buy my Kotlin Cookbook