

【Kotlin】enum class

宣告1: 最基礎的 enum 使用逗號分開

```
enum class DayOfWeek {  
    SUNDAY,  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY,  
    SATURDAY  
}
```

通常搭配when做狀態或是類型判斷

```
val day = DayOfWeek.WEDNESDAY  
when(day) {  
    DayOfWeek.SUNDAY -> println("星期日，猴子刷油漆")  
    DayOfWeek.MONDAY -> println("星期一，猴子添新衣")  
    DayOfWeek.TUESDAY -> println("星期二，猴子肚子餓")  
    DayOfWeek.WEDNESDAY -> println("星期三，猴子去爬山")  
    DayOfWeek.THURSDAY -> println("星期四，猴子去考試")  
    DayOfWeek.FRIDAY -> println("星期五，猴子去跳舞")  
    DayOfWeek.SATURDAY -> println("星期六，猴子去斗六")  
}  
// 星期三，猴子去爬山
```

宣告2: 可包含參數和屬性

例如，我們可以為每一天添加一個參數來表示它是否是工作日：

```
enum class DayOfWeek(val isWeekday: Boolean) {  
    SUNDAY(false),  
    MONDAY(true),  
    TUESDAY(true),  
    WEDNESDAY(true),  
    THURSDAY(true),  
    FRIDAY(true),  
    SATURDAY(false)  
}
```

在這個例子中，每個枚舉常量都覆寫了 `activity` 方法來表示該天的主要活動

```
enum class DayOfWeek(val isWeekday: Boolean) {  
    SUNDAY(false) {  
        override fun activity() = "Relaxing"  
    },  
    MONDAY(true) {  
        override fun activity() = "Working"  
    },  
    TUESDAY(true) {  
        override fun activity() = "Working"  
    },  
    WEDNESDAY(true) {  
        override fun activity() = "Working"  
    },  
    THURSDAY(true) {  
        override fun activity() = "Working"  
    },  
    FRIDAY(true) {  
        override fun activity() = "Working"  
    },  
    SATURDAY(false) {
```

```

        override fun activity() = "Relaxing"
    };

    abstract fun activity(): String
}

println("Today is a weekday: ${today.isWeekday}")
println("Today's activity: ${today.activity()}")

for (day in DayOfWeek.values()) {
    println("${day.name}: Weekday = ${day.isWeekday}, Activity = ${day.activity()}")
}

```

判斷時可以使用enum本身，或是其屬性

```

enum class EDeviceType(val id: String, val label: String) {
    ANDROID("0", "安卓"),
    IOS("1", "iOS");

    companion object {
        fun sortedList(desc: Boolean): List<EDeviceType> {
            return if (desc) {
                entries.sortedByDescending { it.id }
            } else {
                entries.sortedBy { it.id }
            }
        }

        fun ofId(id: String): EDeviceType? {
            return entries.firstOrNull { it.id == id }
        }
    }
}

```

```

val device = EDeviceType.IOS

// 判斷
when(device.id){
    "0" -> println("ANDROID")
    "1" -> println("IOS")
    else -> println("other")
}

when(device.label){
    "安卓" -> println("ANDROID")
    "iOS" -> println("IOS")
    else -> println("other")
}

when(device){
    EDeviceType.IOS -> println("IOS")
    EDeviceType.ANDROID -> println("ANDROID")
    else -> println("other")
}

println(EDeviceType.ofId("0")) //ANDROID
println(EDeviceType.sortedList(true)) //[IOS, ANDROID]

```

enum 可撰寫方法擴充實用度

```

enum class ShowStatus(val id: String, val title: String, private val isPreserve: Boolean = false) {
    // 馬上開播
    TEMP("temp", "馬上開播暫存"),
    ON_AIR("onAir", "馬上開播直播中"),
    PUBLISHED("published", "馬上開播已播出"),
    CANCELED("canceled", "馬上開播前user壓取消"),
}

```

```

STREAM_ENDED("streamEnded", "馬上開播因斷流觸發停播"),

// 預約開播
PRESERVED("preserved", "預約開播已預約", true),
PRESERVED_ON_AIR("preservedOnAir", "預約開播直播中", true),
PRESERVED_PUBLISHED("preservedPublished", "預約開播已播出", true),
PRESERVED_CANCELED("preservedCanceled", "預約開播逾時未開播觸發排程壓取消或user壓刪除", true),
PRESERVED_STREAM_ENDED("preservedStreamEnded", "預約開播因斷流觸發停播", true),

// 違規檔次
BAN("ban", "因違規停播", false);

fun isPlayable(): Boolean {
    return this == TEMP || this == PRESERVED
}

fun isStoppable(): Boolean {
    return this == ON_AIR || this == PRESERVED_ON_AIR
}

fun isStopped(): Boolean {
    return this == PUBLISHED || this == PRESERVED_PUBLISHED
        || this == STREAM_ENDED || this == PRESERVED_STREAM_ENDED
}

fun isBan(): Boolean {
    return this == BAN
}

fun toOnAir(): ShowStatus {
    return if (isPreserve) PRESERVED_ON_AIR else ON_AIR
}

fun toPublished(): ShowStatus {
    return if (isPreserve) PRESERVED_PUBLISHED else PUBLISHED
}

fun toCanceled(): ShowStatus {
    return if (isPreserve) PRESERVED_CANCELED else CANCELED
}

fun toStreamEnded(): ShowStatus {
    return if (isPreserve) PRESERVED_STREAM_ENDED else STREAM_ENDED
}

fun isCanceled(): Boolean {
    return this == CANCELED || this == PRESERVED_CANCELED
}

fun isCancelable(): Boolean {
    return this == TEMP || this == PRESERVED
}

companion object {
    fun ofId(id: String?): ShowStatus = entries.firstOrNull { it.id == id } ?: TEMP

    /**
     * 已開播狀態
     */
    fun ofPublished(): List<ShowStatus> = listOf(
        PUBLISHED, PRESERVED_PUBLISHED, STREAM_ENDED, PRESERVED_STREAM_ENDED,
    )

    /**
     * 開播中狀態
     */
    fun ofOnAir(): List<ShowStatus> = listOf(
        ON_AIR, PRESERVED_ON_AIR,
    )
}

```

```
)

/**
 * 待開播狀態
 */
fun ofTemp(): List<ShowStatus> = listOf(
    TEMP, PRESERVED
)
}
}
```

enum 列舉,取值

```
enum class RGB { RED, GREEN, BLUE }
```

```
fun main() {
    for (color in RGB.entries) println(color) // prints RED, GREEN, BLUE
    println("The first color is: ${RGB.valueOf("RED")}") // prints "The first color is: RED"
}
```

```
enum class RESPONSE( val state: Int) {
    SUCCESS(200),
    NOT_FOUND(404),
    SERVER_ERROR(500),
}
```

```
fun main{
    for (response in RESPONSE.entries) {
        println("Response: ${response.name}, State: ${response.state}, Ordinal: ${response.ordinal}")
    }
}
```

```
// Response: SUCCESS, State: 200, Ordinal: 0
// Response: NOT_FOUND, State: 404, Ordinal: 1
// Response: SERVER_ERROR, State: 500, Ordinal: 2
```

參考：

官方文件 <https://kotlinlang.org/docs/enum-classes.html>

🕒 修訂版本 #11

★ 由 treeman 建立於 9 🕒 2024 17:04:46

✍ 由 treeman 更新於 31 🕒 2024 10:36:13