用 Kotlin 提升你的开发效率

1、data class

```kotlin
data class User(
    val name: String,
    val age: Int
)
```

```java
public class User {

    private String name;
    private int age;

    public User(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public boolean equals(@Nullable Object obj) {
        if (obj instanceof User) {
            User eqUser = (User) obj;
            return this.name.equals(eqUser.name) && this.age == eqUser.age;
        } else {
            return false;
        }
    }
}
```

2、中缀表达式

```kotlin
fun test1() {
    val jack = User( name: "Jack",  age: 20)
    val tom = User( name: "Tom",  age: 20)
    val sameAge = jack sameAge tom
    println(sameAge)
}

/** data class ...*/
data class User(
        val name: String,
        val age: Int)

/** 中缀表达式 ...*/
infix fun User.sameAge(user: User): Boolean {
    return this.age == user.age
}
```

3、标准函数

```
fun test2() {
    val json = """{ ... """.trimIndent()

    val jsonObject = try {
        JSONObject(json)
    } catch (e: JSONException) {
        println("---> error: ${e.message}")
        null
    }
    dealCityInfo(jsonObject) {
        println("---> failed.")
    }
}

/** 标准函数 ...*/
fun dealCityInfo(data: JSONObject?, fail: () -> Unit) {
    data?.takeIf { it.has( name: "city_info") }
            ?.takeIf { it: JSONObject
                with(it.getJSONObject( name: "city_info")) { this: JSONObject!
                    return@takeIf has( name: "title") && has( name: "data")
                }
            }
            ?.let { it.getJSONObject( name: "city_info") }
            ?.apply { this: JSONObject
                // do something
                println("---> do something.")
            } ?: fail()
}
```

4、扩展方法

```kotlin
/**
 * 扩展POST
 */
fun test4(activity: Activity) {
    activity.apply { this: Activity
        main {
            // main thread.
            println("---> main: ${Thread.currentThread().name}")
            worker {
                // worker thread.
                println("---> worker: ${Thread.currentThread().name}")
            }
            println("---> main2: ${Thread.currentThread().name}")
        }
    }
}

fun Activity.main(todo: () -> Unit) {
    Handler().post {
        todo()
    }
}

fun Activity.worker(todo: () -> Unit) {
    Executors.newSingleThreadExecutor()
            .execute {
                todo()
            }
}
```

5、IO

```
fun javaIO() {
    var br: BufferedReader? = null
    try {
        br = BufferedReader(FileReader(File( pathname: "readme.md")))
        var line: String
        while (true) {
            line = br.readLine() ?: break
            println("---> $line")
        }
    } catch (ignore: Exception) {
    } finally {
        try {
            br?.close()
        } catch (ignore: IOException) {

        }
    }
}
// 快乐的IO
fun luckyIO() {
    BufferedReader(FileReader(File( pathname: "readme.md"))).use { it: BufferedReader
        var line: String
        while (true) {
            line = it.readLine() ?: break
            println("---> $line")
        }
    }

    // 更加精简
    File( pathname: "readme.md").readLines().forEach(::println)
}
```

6、集合类操作

```
/**
 * 集合类操作
 */
fun test6() {
    val numList = intArrayOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
    val numList2 = arrayOf(1, 2, 3)
    val numList3 = Array<Int>( size: 1) { it }
    // 只要有一个满足即成立
    val resultAny = numList.any { it > 5 }
    println("---> resultAny: $resultAny")

    // 所有满足才成立
    val resultAll = numList.all { it > 5 }
    println("---> resultAll: $resultAll")
}
```

7、Sequence

```kotlin
/**
 * Sequence
 *
 * 1、当不需要中间操作时，使用List
 * 2、当仅仅只有map操作时，使用sequence
 * 3、当仅仅只有filter操作时，使用List
 * 4、如果末端操作是first时，使用sequence
 * 5、对于没有提及的其他操作符或者其他操作符的组合，请尝试使用例子去验证一下
 */
fun testSequence() {
    var time = System.currentTimeMillis()
    val list = (1..65535)
            .toList()
            .map { it * 2 }
            .filter { it % 3 == 0 }
    list.first()
    // 76ms
    println("---> list: ${System.currentTimeMillis() - time}")

    time = System.currentTimeMillis()
    val sequence = (1..65535)
            .asSequence()
            .map { it * 2 }
            .filter { it % 3 == 0 }
    sequence.first()
    // 5ms
    println("---> sequence:  ${System.currentTimeMillis() - time}")
}
```