Обработка ошибок в Coroutines

Роман Аймалетдинов

raymaletdin

github.com/Evleaps

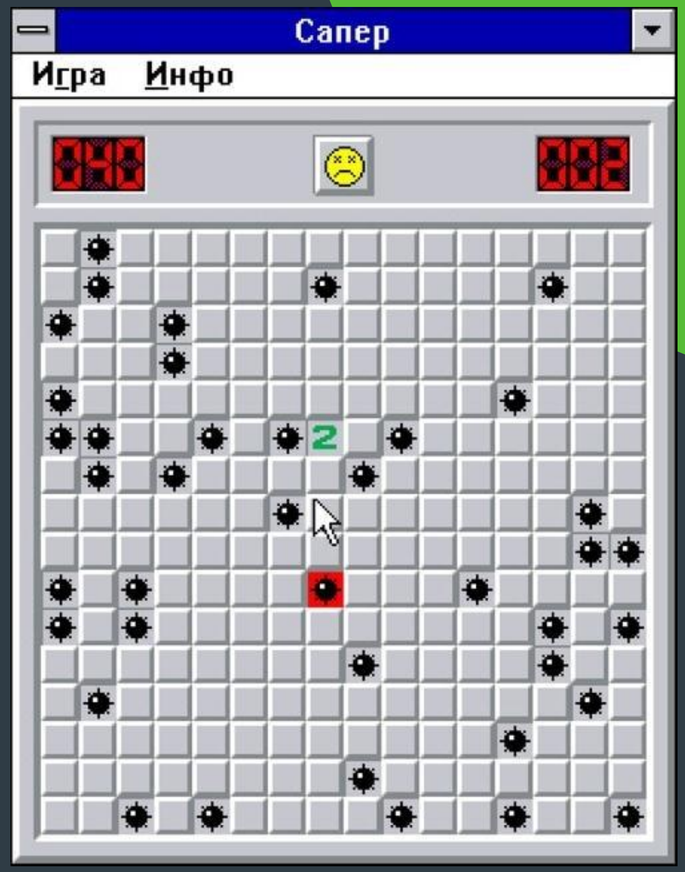
@Evleaps

Содержание доклада

- Как можно обрабатывать ошибки?
- Какие проблемы могут возникнуть?
- Как обезопасить команду?
- Тестирование

Для кого?

- Нет опыта в корутинах
- Команда очень большая
- Все привыкли к Rx
- Нет эксперта, который будет все ревьюить



Как можно обрабатывать ошибки?

Существующие способы

Что-то про CoroutineExceptionHandler

Что-то про try-catch

Flow

Scope

try-catch

try-catch

```
private fun foo() {
    try {
        launch(Dispatchers.IO) {
            loadSmthFromServer()
        }
    } catch (e: Exception) {
        Log.e("ROMAN","Ошибка перехвачена: $e")
    }
}

private fun mapServerResponse(data: ResponseDto): Response {
    if (data.value == null) {
        throw AssertionError("value is null")
    }

    return Response(data.value)
}
```

try-catch

```
private fun foo() {
    try {
        launch(Dispatchers.IO) {
            loadSmthFromServer()
        }
    } catch (e: Exception) {
        Log.e("ROMAN","Ошибка перехвачена: $e")
    }
}

private fun mapServerResponse(data: ResponseDto): Response {
    if (data.value == null) {
        throw AssertionError("value is null")
    }

    return Response(data.value)
}
```

try-catch

```
private fun foo() {
    launch(Dispatchers.IO) {
        try {
            loadSmthFromServer()
        } catch (e: Exception) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}

private fun mapServerResponse(data: ResponseDto): Response {
    if (data.value == null) {
        throw AssertionError("value is null")
    }

    return Response(data.value)
}
```

try-catch

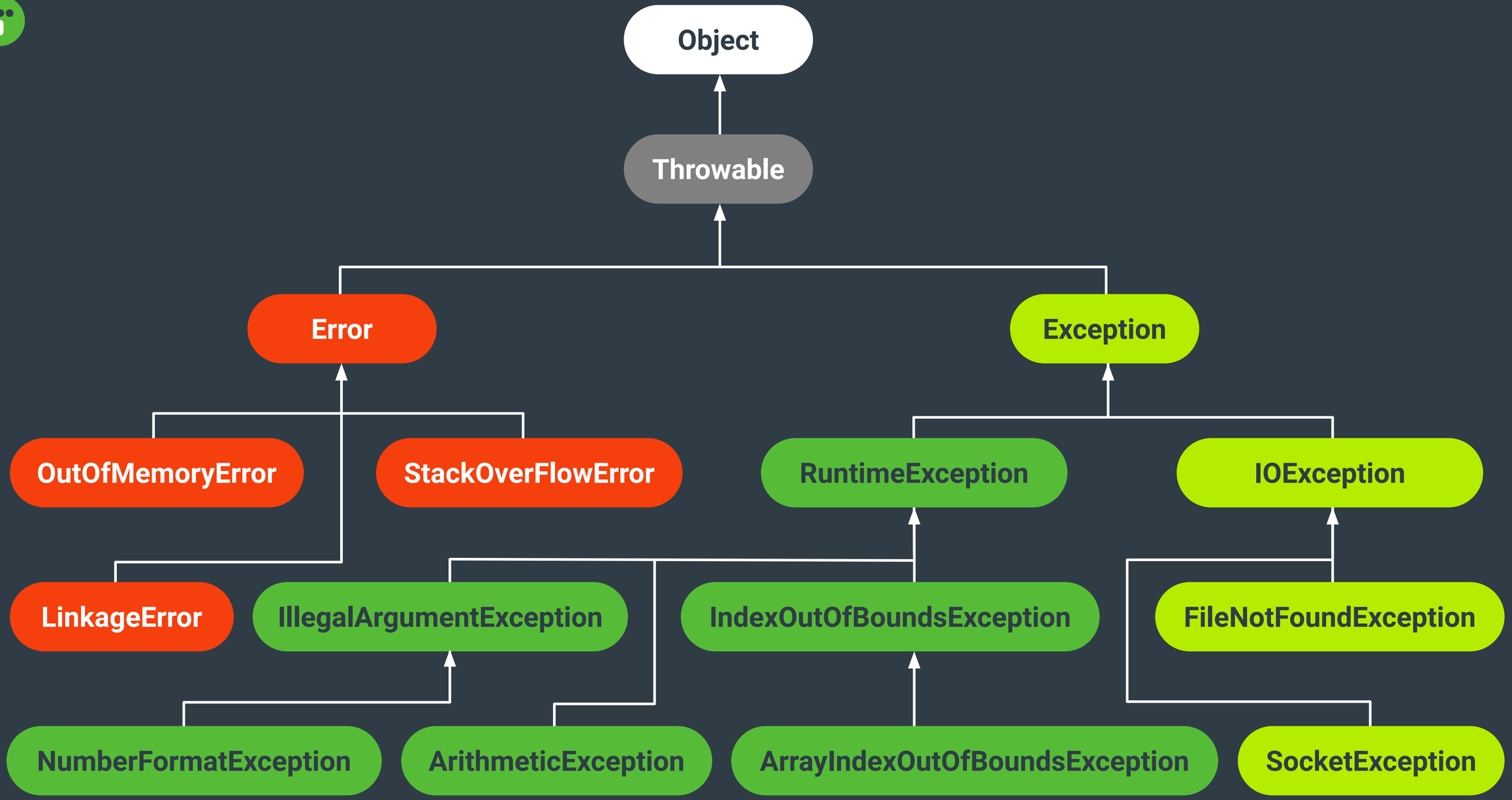
```
private fun foo() {
    launch(Dispatchers.IO) {
        try {
            loadSmthFromServer()
        } catch (e: Exception) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}

private fun mapServerResponse(data: ResponseDto): Response {
    if (data.value == null) {
        throw AssertionError("value is null")
    }

    return Response(data.value)
}
```

try-catch

```
private fun foo() {
    launch {
        try {
            throw IllegalAccessException("ОШИБКА")
        } catch (e: Exception) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}
```


Object
Throwable
Error
Exception
OutOfMemoryError
StackOverFlowError
RuntimeException
IOException
LinkageError
IllegalArgumentException
IndexOutOfBoundsException
FileNotFoundException
NumberFormatException
ArithmeticException
ArrayIndexOutOfBoundsException
SocketException

try-catch

```
private fun foo() {
    launch(Dispatchers.IO) {
        try {
            loadSmthFromServer()
        } catch (e: Throwable) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}

private fun mapServerResponse(data: ResponseDto): Response {
    if (data.value == null) {
        throw AssertionError("value is null")
    }

    return Response(data.value)
}
```

try-catch

```
private fun foo() {
    launch(Dispatchers.IO) {
        try {
            loadSmthFromServer()
        } catch (e: Throwable) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}

private fun loadSmthFromServer() {
    launch(Dispatchers.IO) {
        throw IOException()
    }
}
```

try-catch

```
private fun foo() {
    launch(Dispatchers.IO) {
        try {
            loadSmthFromServer()
        } catch (e: Throwable) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}

private fun loadSmthFromServer() {
    launch(Dispatchers.IO) {
        throw IOException()
    }
}
```

try-catch

```
private fun foo() {
    launch(Dispatchers.IO) {
        try {
            loadSmthFromServer()
        } catch (e: Throwable) {
            Log.e("ROMAN","Ошибка перехвачена: $e")
        }
    }
}

suspend fun loadSmthFromServer(scope: CoroutineScope): List<Any> = scope
    .async {
        delay(100)
        throw IOException()
    }
    .await()
```

try-catch

https://stackoverflow.com › ex... ▾ Перевести эту страницу

Exception not being caught in Coroutines - Stack Overflow

22 мая 2019 г. — **try/catch** block **doesn't** normally **work** in **coroutines** that way. That's why we need to pass context as CoroutineExceptionHandler to handle it **works** ...

3 ответа · 1 голос: Another way to resolve this would be to covert your custom error object to i…

Why i can't use **try/catch** to catch **exception** in Kotlin **coroutine**?	17 сент. 2020 г.
Kotlin **coroutine exception** handling - how to abstract the try ...	7 нояб. 2019 г.
Kotlin: **try catch** in CoroutineScope still crash the app - Stack ...	12 февр. 2022 г.
CoroutineExceptionHandler does not catch **exception** - Stack ...	25 мар. 2022 г.

Другие результаты с сайта stackoverflow.com

Flow

```
flow {
    repeat(5) {
        if (it == 3) throw AssertionError()
        emit("Значение $it")
    }
}.catch {
    println("Ошибочка вышла")
}.collect {
    println(it)
}
```

CoroutineExceptionHandler

CoroutineExceptionHandler

- А-ля “Thread.uncaughtExceptionHandler”
- Ловит только неперехваченные исключения
- На момент вызова корутина уже завершена

```
val exHandler = CoroutineExceptionHandler { coroutineContext, throwable →
    Log.d("ROMAN", "Перехватили: $throwable")
}

private fun foo() {
    launch(Dispatchers.IO) {
        loadSmthFromServer()
    }
}

private fun loadSmthFromServer() {
    launch(Dispatchers.IO) {
        throw IOException()
    }
}
```

```
val exHandler = CoroutineExceptionHandler { coroutineContext, throwable →
    Log.d("ROMAN", "Перехватили: $throwable")
}
private val viewModelContext: Job = Job()
override val coroutineContext = viewModelContext + Dispatchers.Main + exHandler


private fun foo() {
   launch(Dispatchers.IO) {
       delay(3000)
       throw AssertionError()
   }

   launch(Dispatchers.IO) {
       delay(5000)
   }
}
```

```
val exHandler = CoroutineExceptionHandler { coroutineContext, throwable →
    Log.d("ROMAN", "Перехватили: $throwable")
}
private val viewModelContext: Job = Job()
override val coroutineContext = viewModelContext + Dispatchers.Main + exHandler


private fun foo() {
    launch(Dispatchers.IO) {
        delay(3000)
        Log.d("ROMAN", "Отработает")
        throw AssertionError()
    }

    launch(Dispatchers.IO) {
        delay(5000)
        Log.d("ROMAN", "Не отработает")
    }
}
```

```
val exHandler = CoroutineExceptionHandler { coroutineContext, throwable →
    Log.d("ROMAN", "Перехватили: $throwable")
}
private val viewModelContext: Job = SupervisorJob()
override val coroutineContext = viewModelContext + Dispatchers.Main + exHandler


private fun foo() {
   launch(Dispatchers.IO) {
       delay(3000)
       Log.d("ROMAN", "Отработает")
       throw AssertionError()
   }

   launch(Dispatchers.IO) {
       delay(5000)
       Log.d("ROMAN", "Отработает")
   }
}
```

```
val exHandler = CoroutineExceptionHandler { coroutineContext, throwable →
    Log.d("ROMAN", "Перехватили: $throwable")
}
private val viewModelContext: Job = SupervisorJob()
override val coroutineContext = viewModelContext + Dispatchers.Main + exHandler


private fun foo() {
   launch(Dispatchers.IO) {
       throw AssertionError("Нужен тост")
   }
   launch(Dispatchers.IO) {
      throw AssertionError("Нужен диалог")
   }
   launch(Dispatchers.IO) {
       throw AssertionError("Только запись в лог")
   }
}
```

```
val exHandler = CoroutineExceptionHandler { coroutineContext, throwable →
    when (throwable) {
        is MappingException → {}
        is DebugException → {}
        is MyFeatureException → {}
        is MyOtherFeatureException → {}
        is MyFeatureException2 → {}
    }
}
```

```
val exHandler1 = CoroutineExceptionHandler { _, throwable →
    // перейти на другой экран
}
val exHandler2 = CoroutineExceptionHandler { _, throwable →
    // Показать тост
}


private fun foo() {
    launch(Dispatchers.IO + exHandler1) {
        throw AssertionError("Нужен переход на спец. экран")
    }

    launch(Dispatchers.IO + exHandler2) {
        throw AssertionError("Нужен тост")
    }
}
```

```
class MyViewModel : CoroutineScope {

    private val exHandler = CoroutineExceptionHandler { _, _ →
        view.showError() // UI changing
    }

    private val job: Job = SupervisorJob()
    override val coroutineContext = job + Dispatchers.Main

    private fun showCoupons() {
        launch(context = exHandler + Dispatchers.IO) {
            val coupons = promoInteractor.loadCoupons()

            withContext(Dispatchers.Main) {
                view.showCoupons(coupons)
            }
        }
    }
}
```

```
class MyViewModel : CoroutineScope {

    private val exHandler = CoroutineExceptionHandler { _, _ →
        view.showError() // CRASH!
    }

    private val job: Job = SupervisorJob()
    override val coroutineContext = job + Dispatchers.Main + exHandler

    private fun showCoupons() {
        launch(context = exHandler + Dispatchers.IO) {
            val coupons = promoInteractor.loadCoupons()

            withContext(Dispatchers.Main) {
                view.showCoupons(coupons)
            }
        }
    }
}
```

```
class MyViewModel : CoroutineScope {

    private val exHandler = CoroutineExceptionHandler { _, _ ->

        withContext(Dispatchers.Main) {
            view.showError() // UI changing
        }
    }

    private val job: Job = SupervisorJob()
    override val coroutineContext = job + Dispatchers.Main + exHandler

    private fun showCoupons() {
        launch(context = exHandler + Dispatchers.IO) {
            val coupons = promoInteractor.loadCoupons()

            withContext(Dispatchers.Main) {
                view.showCoupons(coupons)
            }
        }
    }
}
```

```
private val exHandler = CoroutineExceptionHandler { _, exception →
      Log.d("ROMAN", "New exception $exception")
}

private val job: Job = SupervisorJob()
override val coroutineContext = job + Dispatchers.Main + exHandler

private fun foo() {
   launch {
       launch {
           delay(1000)
           throw ArithmeticException() // второе исключение
       }

       launch {
           delay(100)
           throw IOException() // первое исключение
       }
   }
}
```

```
private val exHandler = CoroutineExceptionHandler { _, exception →
      Log.d("ROMAN", "New exception $exception")
}

private val job: Job = SupervisorJob()
override val coroutineContext = job + Dispatchers.Main + exHandler

private fun foo() {
   launch {
       launch {
           delay(1000)
           throw ArithmeticException() // второе исключение не вызовется
       }

       launch {
           delay(100)
           throw IOException() // первое исключение
       }
   }
}
```

Сложно и неочевидно.
Нужна хорошая память!

- Слабый контракт
- try-catch ненадежен
- Вложенность
- Ошибка приходит в UI потоке
- Можно уронить соседнюю корутину
- Не обязывает обработать факт ошибки

```
fun showCoupons() {
    launch(Dispatchers.IO) {
        val coupons = loadCouponsFromCache()
        show(coupons)
    }
}
```

```
fun showCoupons() {
    launchIO(
        safeAction = {
            val coupons = loadCouponsFromCache()
            show(coupons)
        },
        onError = Timber::e
    )
}
```

```
fun showCoupons() {
    launchIO(
        safeAction = {
            val coupons = loadCouponsFromCache()
            show(coupons)
        },
        onError = { /* do nothing */ }
    )
}
```

Обработка ошибок от Кирилла Розова


2 стула
Все должны выучить все нюансы
Быть готовым к ошибкам команды
Усиленно онбордить новичков
Продумать код стайл
Сделать ограничения
Смириться с урезанной функциональностью

SafeCoroutines

- launch(Dispatchers.IO) → launchIO
- launch(Dispatchers.Main) → launchMain
- withContext(Dispatchers.IO) → withIO
- withContext(Dispatchers.Main) → withMain

```
inline fun CoroutineScope.launchIO(
    crossinline safeAction: suspend () -> Unit,
    crossinline onError: (Throwable) -> Unit
): Job {
    val exceptionHandler = CoroutineExceptionHandler { _, throwable ->
        launch(Dispatchers.Main) {
            onError.invoke(throwable)
        }
    }

    return this.launch(exceptionHandler + Dispatchers.IO) {
        safeAction.invoke()
    }
}
```

```
inline fun CoroutineScope.launchIO(
    crossinline safeAction: suspend () -> Unit,
    crossinline onError: (Throwable) -> Unit,
    errorDispatcher: CoroutineDispatcher = Dispatchers.Main
): Job {
    val exceptionHandler = CoroutineExceptionHandler { _, throwable ->
        launch(errorDispatcher) {
            onError.invoke(throwable)
        }
    }

    return this.launch(exceptionHandler + Dispatchers.IO) {
        safeAction.invoke()
    }
}
```

```
inline fun CoroutineScope.launchMain(
    crossinline safeAction: suspend () -> Unit,
    crossinline onError: (Throwable) -> Unit
): Job {
    val exceptionHandler = CoroutineExceptionHandler { _, throwable ->
        launch(Dispatchers.Main) {
            onError.invoke(throwable)
        }
    }

    return this.launch(exceptionHandler + Dispatchers.Main) {
        safeAction.invoke()
    }
}
```

```
@Suppress("NeedToUseCustomWithContextRule")
suspend inline fun <T> withIO(noinline block: suspend CoroutineScope.() → T): T {
    return withContext(Dispatchers.IO, block)
}

@Suppress("NeedToUseCustomWithContextRule")
suspend inline fun <T> withMain(noinline block: suspend CoroutineScope.() → T): T {
    return withContext(Dispatchers.Main, block)
}
```

Было

```
class MyViewModel : CoroutineScope {

    private val exHandler = CoroutineExceptionHandler { _, _ ->

        withContext(Dispatchers.Main) {
            view.showError() // UI changing
        }
    }

    private val job: Job = SupervisorJob()
    override val coroutineContext = job + Dispatchers.Main + exHandler

    private fun showCoupons() {
        launch(context = exHandler + Dispatchers.IO) {
            val coupons = promoInteractor.loadCoupons()

            withContext(Dispatchers.Main) {
                view.showCoupons(coupons)
            }
        }
    }
}
```

Стало

```
class MyViewModel : CoroutineScope {

    private val job: Job = SupervisorJob()
    override val coroutineContext = job

    private fun showCoupons() {
        launchIO(
            safeAction = {
                val coupons = promoInteractor.loadCoupons()

                withMain {
                    view.showCoupons(coupons)
                }
            },
            onError = view::showError
        )
    }
}
```

Кратко

```
class MyViewModel : CoroutineScope {

    private val job: Job = SupervisorJob()
    override val coroutineContext = job

    private fun showCoupons() {
        launchIO({
            val coupons = promoInteractor.loadCoupons()
            withMain { view.showCoupons(coupons) }
        }, view::showError)
    }
}
```

- Жесткий контракт
- Знакомый из Rx синтаксис
- Проще код
- Очевидность обработки
- Ошибки всегда в Main
- Изолированные ошибки

Мудрости

Легко сказать — разберись с корутинами. Но точно ли это сможет вся ваша команда?

Detekt

```kotlin
internal class NeedToUseCustomLaunchRule(config: Config) : Rule(config) {

    companion object {
        private const val TRIGGER_VALUE = ".launch {"
    }

    override val issue = Issue(
        id = "NeedToUseCustomLaunch",
        description = "Must use a custom launch ext from CoroutinesUtils instead \".launch { smth }\"",
        severity = Severity.CodeSmell,
        debt = Debt.FIVE_MINS
    )

    override fun visitNamedFunction(function: KtNamedFunction) {
        super.visitNamedFunction(function)

        var offset = 0
        val lines = function.text.lines()

        for (line in lines) {
            offset += line.length
            if (line.contains(TRIGGER_VALUE)) {
                report(
                    CodeSmell(
                        issue = issue,
                        entity = Entity.from(function, offset),
                        message = "The function ${function.name} using Coroutines. " +
                            "You must use a custom launch ext from CoroutinesUtils instead \".launch { smth }\" here."
                    )
                )
            }

            offset += 1 // '\n'
        }
    }
}
```

СИТИМОБИЛ

SafeCoroutines

ссылка на ресурс:
https://github.com/Evleaps/SafeCoroutines.git

Тестирование

DI || coroutines-test

```
interface CoroutineDispatchers {

    fun io(): CoroutineDispatcher
    fun ui(): CoroutineDispatcher
}



class AppDispatcher @Inject constructor(): CoroutineDispatchers {

    override fun io() = Dispatchers.IO
    override fun ui() = Dispatchers.Main
}



class TestDispatcher: CoroutineDispatchers {

    override fun io() = Dispatchers.Unconfined
    override fun ui() = Dispatchers.Unconfined
}
```

```
class GetAllBooksAndAuthorsUseCase @Inject constructor(
    val booksRepository: BooksRepository,
    val authorsRepository: AuthorsRepository,
    val appDispatcher: CoroutineDispatchers
) {

    suspend fun getBookAndAuthors(): BookAndAuthors {
        return coroutineScope {
            val books = async(appDispatcher) {
                booksRepository.getAllBooks()
            }

            val authors = async(appDispatcher) {
                authorsRepository.getAllAuthors()
            }

            BookAndAuthors(books.await(), authors.await())
        }
    }
}
```

```
inline fun CoroutineScope.launchSafe(
    crossinline safeAction: suspend () -> Unit,
    crossinline onError: (Throwable) -> Unit,
    dispatcher: CoroutineDispatcher,
    errorDispatcher: CoroutineDispatcher = Dispatchers.Main,
): Job {
    val exceptionHandler = CoroutineExceptionHandler { _, throwable ->
        launch(errorDispatcher) {
            onError.invoke(throwable)
        }
    }

    return this.launch(exceptionHandler + dispatcher) {
        safeAction.invoke()
    }
}
```

```
class MyViewModel @Inject constructor(
    val appDispatcher: CoroutineDispatchers
) {

    fun showCoupons() {
        launchSafe(
            safeAction = {
                doWork()
            },
            onError = Timber::e,
            dispatcher = appDispatchers.io
    )

}
```

```
class MyViewModel @Inject constructor(
    val appDispatcher: CoroutineDispatchers
) {

  fun runLaunchBuilder(): Job {
      return launchBuilder()
          .launchOn(Dispatchers.IO)
          .errorOn(Dispatchers.Main)
          .onError(this::onError)
          .launch {
              // some logic
          }
  }
}
```

coroutines-test

```
testImplementation ‘org.jetbrains.kotlinx:kotlinx-coroutines-test:1.6.2’
```

```
@Before
fun setUp() {
    Dispatchers.setMain(StandardTestDispatcher())
}


@After
fun tearDown() {
    Dispatchers.resetMain()
}
```

```
fun runSmthInIO(): Job {
    return launchIO(
        safeAction = {
            delay(3_000L)
            onSuccess()
        },
        onError = ::onError
    )
}


@Test
fun `test_name`() = runTest {
    val viewModel: LikeViewModel = spy(LikeViewModel())

    viewModel.runSmthInIO().join()
    verify(viewModel, times(1)).onSuccess()
}
```

```
fun runSmthInIO(): Job {
    return launchIO(
        safeAction = {
            delay(3_000L)
            onSuccess()
        },
        onError = ::onError
    )
}


@Test
fun `test_name`() = runTest {
    val viewModel: LikeViewModel = spy(LikeViewModel())

    viewModel.runSmthInIO().join()
    verify(viewModel, times(1)).onSuccess()
}
```

```
private var isFlagEnabled = false

suspend fun checkThatFlagTrue(): Boolean {
    withIO {
        delay(3_000)
        isFlagEnabled = true
    }
    return isFlagEnabled
}


@Test
fun `test_name`() = runTest {
    val mainViewModel = LikeViewModel()

    val isTrue = mainViewModel.checkThatFlagTrue()
    assert(isTrue)
}
```

```kotlin
@Test
fun `test_name`() = runTest {
    val repository = mutableListOf<String>()

    launchMain(
        safeAction = { repository.add("Joshua") },
        onError = { /* nothing */ }
    )

    launchMain(
        safeAction = { repository.add("Roman") },
        onError = { /* nothing */ }
    )

    assertEquals(listOf("Joshua", "Roman"), repository)
}
```

```
@Test
fun `test_name`() = runTest {
    val repository = mutableListOf<String>()

    launchMain(
        safeAction = { repository.add("Joshua") },
        onError = { /* nothing */ }
    )

    launchMain(
        safeAction = { repository.add("Roman") },
        onError = { /* nothing */ }
    )

    advanceUntilIdle()

    assertEquals(listOf("Joshua", "Roman"), repository)
}
```

inDriver

InDriver.Tech

100 млн

установок приложения

40+

стран

4 млн

сделок в сутки

Роман Аймалетдинов

raymaletdin

github.com/Evleaps

@Evleaps