

Advanced Kotlin

Матвей Попов

План

- Многопоточность в глобальном контексте
- Многопоточность в контексте Java
- Многопоточность в контексте Kotlin

- Kotlin Coroutines, live coding
- Сборщики проектов
- Gradle: зачем и почему
- Домашнее задание

Краткий экскурс в МНОГОПОТОЧНОСТЬ

Основные определения

Многозадачность

Свойство **операционной** системы или среды выполнения обеспечивать возможность **параллельной** (или псевдопараллельной) обработки нескольких задач.

Многопоточность

Свойство **платформы** (например, операционной системы, виртуальной машины и т. д.) или приложения, состоящее в том, что **процесс**, порождённый в операционной системе, может состоять из нескольких **потоков**, выполняющихся «**параллельно**», то есть без предписанного порядка во времени

Процессы и потоки

С точки зрения пользователя

**Процесс - экземпляр программы
во время выполнения**

**Потоки - ветви кода,
выполняющиеся «параллельно»**

**С точки зрения операционной
системы**

Процесс - это абстракция,
реализованная на уровне
операционной системы

Процесс - просто контейнер, в котором находятся ресурсы программы

Процесс содержит:

- Адресное пространство
- Потоки
- Открытые файлы
- Дочерние процессы
- И т.д.

Поток - это абстракция,
реализованная на уровне
операционной системы

Поток - просто **контейнер**, в
котором хранится информация о
состоянии выполнения программы

Поток содержит:

- Счетчик команд
- Регистры
- Стек

И в чем же отличия?

**Процесс - заявка на все
виды ресурсов**

**Поток - заявка на
процессорное время**

**Процесс - способ сгруппировать
данные и ресурсы**

**Поток - это единица
выполнения**

А что там есть еще?

- Планирование потоков
- Состояние потоков
- Приоритет потоков
- Системные вызовы
- Режимы доступа

Почему это все важно?

- Понимание работы потоков залог предсказуемой эксплуатации приложения
- Понимание работы потоков залог написания правильного многопоточного кода

НЕ ВСЕГДА

os.Thread == lang.Thread

Многопоточность в Java

Процесс в контексте Java её
среда выполнения: JRE

```
9  ► public class Main {
10
    1 usage
11  ▮ public static class MyThread extends Thread {
12
13      @Override
14  ▮ public void run() {
15          System.out.println("Hello from MyThread");
16      }
17  }
18
19  ► ▮ public static void main(String[] args) throws InterruptedException {
20      Thread myThread = new MyThread();
21
22      System.out.println("Hello from Main thread");
23
24      myThread.start();
25      myThread.join();
26
27      System.out.println("Buy from Main thread");
28  }
29 }
30
```

Java multithreading

- `java.lang.Thread == os.Thread`
- Есть стандартные примитивы синхронизации
- Богатая библиотека `java.util.concurrent`

Все же круто, зачем нам что-то еще?



**А как мы вообще используем
эту вашу **многопоточность**?**

98% 10

Operations

И что же это значит?

- Потоки глобально можно разделить на CPU-based и IO-based
- Почти всегда мы выполняем IO операции
- При выполнении IO операции мы большую часть времени ждем

И ЧТО В ЭТОМ ПЛОХОГО?

- Создание потоков - дорого
- Переключение контекста потоков - дорого
- Обслуживание потоков - дополнительная память
- Управление потоков - дорого
- Написание правильного многопоточного кода - очень сложно

И что же с этим делать?

ThreadPool-ы и ExecutorService-ы

- Создаем какой-то контейнер потоков
- Закидываем задачи куда-то “внутрь”
- Не думаем о thread-management
- Можем гибко конфигурировать и использовать разные стратегии

**А что делать с блокировкой
потоков на IO?**

**Давайте использовать
callback-и!**

Пишите вы код, а потом...

```
public void getCredentialsWithCallback(String login, Handler<AsyncResult<Credentials>> resultHandler) {
    client.prepare( query: """
        SELECT user_id, password FROM app.credentials
        WHERE provider = :provider AND login = :login
        """, statementHandler -> {
        if (statementHandler.succeeded()) {
            client.execute(statementHandler.result().bind().setString( name: "login", login), selectHandler -> {
                if (selectHandler.succeeded()) {
                    selectHandler.result().one(fetchHandler -> {
                        if (fetchHandler.succeeded()) {
                            Row row = fetchHandler.result();
                            if (row == null) {
                                resultHandler.handle(Future.failedFuture(new NoSuchElementException("Wrong login or password")));
                            } else {
                                resultHandler.handle(Future.succeededFuture(new Credentials(login, row.getString( name: "password"), row.getUUID( name: "user_id"))));
                            }
                        } else {
                            resultHandler.handle(Future.failedFuture(fetchHandler.cause()));
                        }
                    });
                } else {
                    resultHandler.handle(Future.failedFuture(selectHandler.cause()));
                }
            });
        } else {
            resultHandler.handle(Future.failedFuture(statementHandler.cause()));
        }
    });
}
```

**А может тогда попробуем
реактивное программирование?**

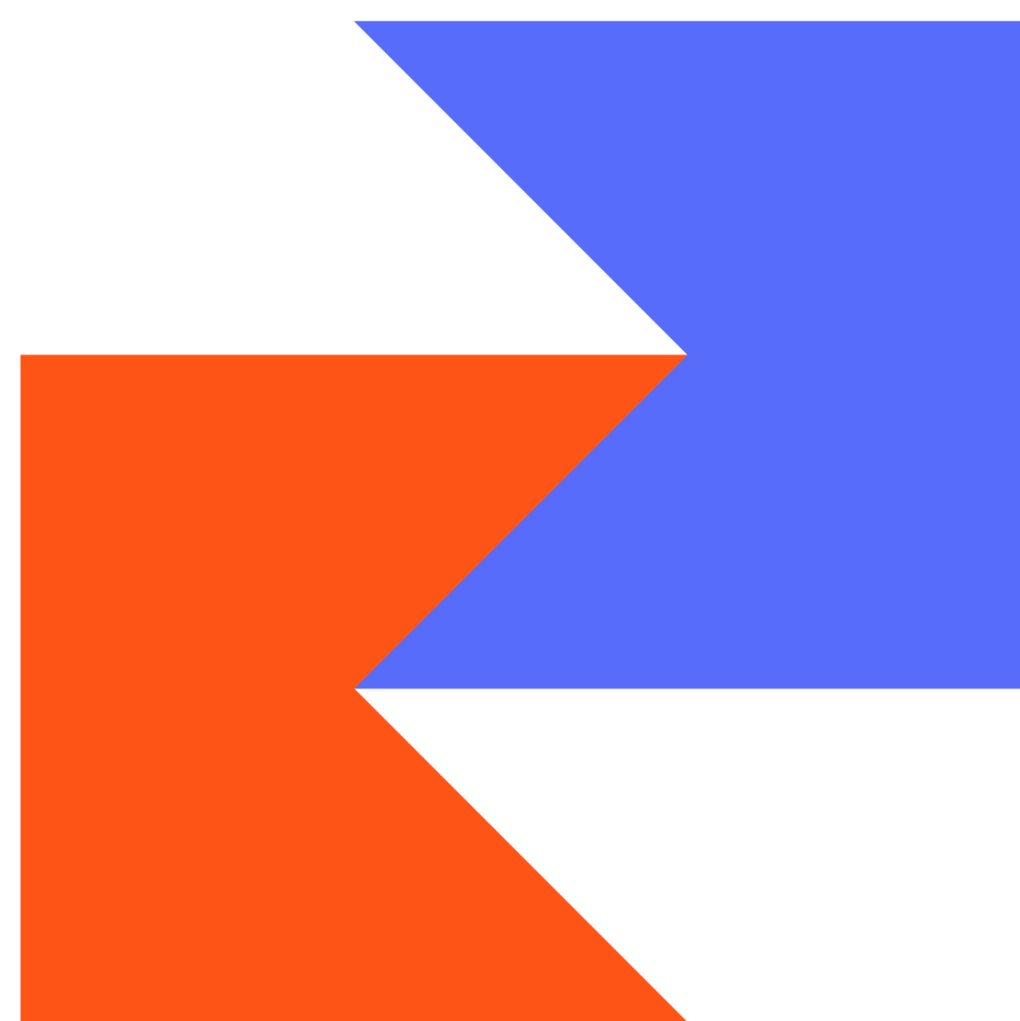
```
public Observable<String> send(String command) {  
    return Observable.just(command)  
        .doOnNext(cmd -> checkConnection())  
        .map(cmd -> cmd.getBytes())  
        .map(bytes -> addHeader(bytes))  
        .map(bytes -> sendBytes(bytes))  
        .timeout(MAX_SEND_TIMEOUT_MS, TimeUnit.MILLISECONDS)  
        .map(result -> readAnswer())  
        .doOnError(throwable -> disconnect())  
        .retry(MAX_RETRY_COUNT)  
        .subscribeOn(Schedulers.io());  
}
```

Ну вроде круто, правда же?

- Нужно хорошо знать API библиотеки
- Каждое действие это некоторый оператор
- Достаточно специфичный подход к написанию кода
- Как интегрироваться с другими системами?

- Решение не из “коробки”
- В разы нагружаем GC

И что же нам тогда делать?



Kotlin

Coroutines

Многопоточность в Kotlin

Kotlin Coroutines

Что значит coroutine?

**Как работают coroutine в
Kotlin?**

**Есть ли что-то похожее в Java
сейчас?**

Выводы

Немного live coding

Системы сборки проектов

Подробнее про Gradle

Как он работает?

**Почему мы будем
использовать Gradle?**

Домашнее задание

Материалы

Спасибо за уделенное время!