УДК 004.422.832:004.438

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ COROUTINES В KOTLIN

Боловинцев А.С., Белячевский М.С., Василькова А.Н.

Белорусский государственный университет информатики и радиоэлектроники,

г. Минск, Республика Беларусь

Научный руководитель: Потапенко Н. И. – ст. преподаватель кафедры ИПиЭ

Аннотация. Асинхронное программирование играет немаловажную роль, оно позволяет увеличить объёмы работы, которую ваше приложение может выполнять одновременно с другими задачами. Coroutines Kotlin представляют новый стиль параллелизма, который можно использовать на Android для упрощения асинхронного кода.

Ключевые слова: coroutines, kotlin, асинхронное программирование.

Введение. В настоящее время Kotlin является одним из самых популярных языков программирования для разработки Android-приложений. Одной из ключевых особенностей Kotlin являются Coroutines - механизм, позволяющий писать асинхронный код более просто и эффективно. Асинхронное программирование является важной частью современного подхода к разработке программного обеспечения. Оно увеличивает объем работы, которую ваше приложение может выполнять параллельно. Это позволяет выполнять сложные задачи вдали от потока пользовательского интерфейса в фоновом режиме, что в конечном итоге обеспечивает плавный и лучший опыт для пользователя.

Основная часть. Язык программирования Kotlin определяет coroutin как "легкие нити". Это своего рода задачи, которые могут выполнять фактические потоки. Coroutines были добавлены в Kotlin в версии 1.3 и основаны на установленных концепциях из других языков. Coroutines Kotlin представляют новый стиль параллелизма, который можно использовать для Android в целях упрощения асинхронного кода.

Coroutines в основном бывают двух типов:

– Stackless;

– Stackful.

Kotlin реализует безслойные coroutines, это означает, что у coroutines нет собственного стека, поэтому они не отображаются в родном потоке. Так почему же нужно узнавать что-то новое?

При использовании Rx требуется много усилий, чтобы получить его достаточно, чтобы использовать его безопасно. С другой стороны, AsyncTasks и потоки могут легко привести к утечкам и накладным расходам на память. Даже при использовании этих инструментов после стольких недостатков код может страдать от обратных вызовов, которые приводят к увеличению объёма кода. Не только это, но и код также становится нечитаемым, так как он имеет много обратных вызовов, которые в конечном итоге замедляют устройство, что приводит к плохому пользовательскому опыту.

Android – это однопоточная платформа, по умолчанию все работает на основном потоке. В Android почти каждое приложение должно выполнять некоторые операции, не связанные с пользовательским интерфейсом, такие как (сетевой вызов, операции ввода-вывода), поэтому, когда концепция coroutines не вводится, программист посвящает эту задачу различным потокам, каждый поток выполняет поставленную ему задачу, когда задача завершена, они возвращают результат в поток пользовательского интерфейса для обновления необходимых изменений. Хотя в Android дана подробная процедура о том, как эффективно выполнять эту задачу, используя лучшие практики использования потоков, эта процедура включает в себя множество обратных вызовов для передачи результата между потоками, которые в конечном итоге вводят тонны кода в наше приложение и время ожидания для возврата результата увеличивается.

На Android каждое приложение имеет основной поток (который обрабатывает все операции пользовательского интерфейса, такие как просмотры рисования и другие взаимодействия с пользователями). Если в этом основном потоке происходит слишком много работы, например, сетевые вызовы, приложения будут зависать или замедляться, что приведет к плохому пользовательскому опыту.

−Coroutines является рекомендуемым решением для асинхронного программирования на Android. Некоторые выделенные особенности coroutines приведены ниже.

−легкий: можно запустить много coroutines на одной резьбе благодаря поддержке подвески, которая не блокирует резьбу, на которой работает coroutine. Приостановки освобождают память от блокировки, поддерживая при этом несколько одновременных операций.

−Встроенная поддержка отмены: Отмена генерируется автоматически через запущенную иерархию coroutine.

−Меньше утечек памяти: он использует структурированный параллелизм для выполнения операций в рамках области видимости.

−Интеграция Jetpack: Многие библиотеки Jetpack включают расширения, которые обеспечивают полную поддержку сопрограмм. Некоторые библиотеки также предоставляют свой собственный объем корутина, который можно использовать для структурированного параллелизма[1].Сопрограммы предоставляют нам простой способ синхронного и асинхронного программирования. Сопрограммы позволяют приостанавливать выполнение и возобновлять его позже в какой-то момент в будущем, что лучше всего подходит для выполнения неблокирующих операций в случае многопоточности [2].

Проблема в использовании coroutine заключается в достаточно громоздком интерфейсе, а также в том, что сопрограммы не наследуют контекст автоматически. Всякий раз, когда вы вызываете конструктор сопрограммы, такой как async, runBlocking или launch, вы по умолчанию теряете текущий контекст сопрограммы.

Однако, на фоне перечисленных выше слабых мест, выделяются их преимущества:

1 Память эффективная или легкая

Есть возможность запустить много coroutines в одном потоке благодаря поддержке функций приостановки, которые не блокируют запуск потока. Это, в свою очередь, потребляет меньше памяти, так как приостановка потока дороже.

2 Структурированный параллелизм

Coroutines обеспечивают поддержку структурированного параллелизма, т.е. любое исключение или отмена внутри coroutine распространяются через иерархию, чтобы с ним можно было правильно обращаться.

3 Меньше утечек памяти.

Поскольку параллелизм распространяется по иерархии, вероятность утечки меньше.

4 Синхронный код.

Запуск coroutine почти как написание синхронного кода, который делает код очень чистым, читаемым и менее подверженным ошибкам.

5 Улучшение контроля в асинхронном процессе

Выполнение асинхронных вычислений в coroutines обеспечивает лучший контроль с точки зрения отмены или тайм-аута, объема coroutine, переключения контекста, основной безопасности и параллелизма.

6 Поддержка Jetpack

Многие библиотеки Jetpack включают расширения, которые обеспечивают полную поддержку корутинов. Некоторые библиотеки также предоставляют свою собственную область coroutine, которую вы можете использовать для структурированного параллелизма[3].

Coroutines всегда выполняется в контексте, который называется CoroutineContext. CoroutineContext — это набор различных элементов, два из которых — Job и Dispatcher.

Конструктор запуска сопрограммы возвращает объект Job, который помогает нам поддерживать жизненный цикл нашей сопрограммы. Мы можем отменить сопрограмму, используя этот объект задания. Отмена родительского задания немедленно отменяет все его дочерние элементы, а отказ дочернего элемента с исключением, отличным от CancellationException, немедленно отменяет его родительский элемент и, следовательно, все его другие дочерние элементы.

CoroutineContext включает диспетчер сопрограмм, который сообщает нам, какой поток сопрограммы использует для своего выполнения. Если сопрограмма запускается без диспетчера, она запускается в диспетчере по умолчанию. Мы можем явно указать диспетчер на сопрограмме.

Заключение. Kotlin Coroutines предоставляют несколько преимуществ перед Async Task and Handlers, включая лучшую производительность, более детальный контроль над управлением потоками, поддержку управления жизненным циклом и расширенные механизмы синхронизации потоков. Эти функции делают Kotlin Coroutines более подходящими для целого ряда задач, включая длительные и высокопараллельные задачи, и помогают предотвратить распространенные проблемы, такие как узкие места в производительности и утечки памяти. Поэтому Kotlin Coroutines - лучший выбор, чем Async Task and Handlers для большинства приложений Android.

Список литературы

1. *Geeksforgeeks [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/kotlin-coroutines-on-android/ – Дата доступа: 02.03.2023.*

2. *Medium [Электронный ресурс]. – Режим доступа: https://proandroiddev.com/kotlin-coroutines-in-andriod-ff0b3b399fa0 – Дата доступа: 02.03.2023.*

3. *Medium [Электронный ресурс]. – Режим доступа: https://blog.devgenius.io/kotlin-coroutines-what-why-how-99529c951a2e – Дата доступа: 02.03.2023.*

UDC 004.422.832:004.438

FEATURES OF USING COROUTINES IN KOTLIN

Bolovintsev A. S., Belyachevsky M. S., Vasilkova A. N.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Potapenko N. I. – senior lecturer of the Department of EPE

Annotation. Asynchronous programming is very important, it helps to increase the amount of work your application can do in parallel. Coroutines Kotlin introduce a new style of concurrency that can be used on Android to simplify asynchronous code.

Keywords: coroutines, kotlin.