# План видеоролика «Scope Functions»

Пункт = смена кадра

1. Добрый день!

Пришло время продолжить цикл полезных видеороликов по Kotlin. Сегодня ролик будет посвящён функциям поля видимости или контекстным функциям.

Контекстные функции или Scope Functions - специальные функции-расширения в языке Kotlin, которые позволяют выстраивать цепочки действий с объектами. Это «синтаксический сахар», позволяющий разработчику гораздо удобнее взаимодействовать с сущностями языка.

1. Дадим определение: Контекстной функцией является функция, целью которой является выполнение конкретного блока кода для конкретного объекта. Всего в стандартной библиотеке Kotlin их 5: let, run, with, apply и also. Отличаются scope- функции способом обращения к объекту контекста и возвращаемым значением.
2. Посмотрим контекстные функции на примерах (буду показывать код в редакторе и результат выполнения)
3. Для чего?

Применение scope функций позволяет разработчику уйти от «линейного» контекста настройки и использования объекта, а также позволяет лаконично использовать другие возможности Kotlin для построения цепочек вызовов или действий, применяемых если объект существует.

1. Семантика применения

Рассмотрим контекстные функции в рамках слов, которые их описывают. Давайте переведем их и посмотрим в контексте перевода.

let – позволять. Запущенная над обьектом, мы как бы позволяем выполниться следующей операции. Т.к. let часто применяется совместно с safe-оператором ?, понятно, что конструкция obj?.let { \*\*\* } значит «если обьект существует позволь коду внутри блока выполнится». Вообще совместное применение с safe-оператором наиболее частое применение let. Внутренние проверки IDE IDEA часто советуют исправить «классическую» проверку на null именно этой конструкцией.

Вообще let конечно можно применять и без safe-оператора, но разработчики языка(ИМХО) задумывали для этих целей run { \*\*\* }. Но ссылка it наверное короче и все юзают let.

run – «запустить, выполнить» в контексте IT и разработки. Именно та команда, вместо которой все используют let. Они обе возвращают результат лямбды, но тут не переопределённая ссылка на объект-параметр this. Но, учитывая то, что run можно использовать без объекта, можно все создать, запустить и настроить и вернуть результат в переменную результат выполнения.

apply – «применить». Тут название говорит само за себя. Берется объект, после запускается apply и донастраивает его нужным образом. Устанавливаются дополнительные значения в переменные, которые отсутствуют в конструкторе или доп. методы инициализации, и после возвращает донастроенный обьект.

also – «кроме того, также». Значит это следующее «пожалуйста, помимо полученного результата нужно сделать это и это». То есть мы говорим программе на необходимость совершения дополнительного действия после того, как получен какой либо результат. Также функция может быть применена для получения отладочной информации или например, в контексте доп. условия, что если результат транзакции успешен, то нужно отправить email клиенту, что его запрос в обработке.

with – «с чем-то». Учитывая, что with(param) имеет параметр в круглых скобках, над которым совершается действие, то семантика данной функции значит что нужно что- то сделать над определённым объектом. И точка.