Kotlin In Action

by Yev Kovalev

BKUG Meetup #4

06/13/2017

AGENDA

Who

What

How to get

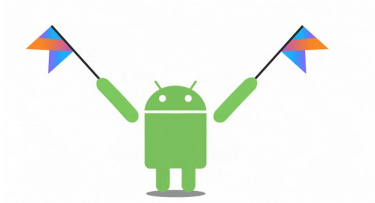
Tricks & advice

Coroutines

Conclusion

- Senior Android developer
- 5+ years of experience (4 years - Android only)
- Originally meant to be J2EE developer (shit happens)
- Some experience in ML, Spring Boot, Vert.x, etc.

- concise
- null safety
- 100% interop with Java (~99%)



IDEA:

- automatically: ‘Configure Kotlin In Project’ option
- manually:
  - gradle: 14 lines

```
buildscript {
    ext {
        kotlin_version = '1.1.2'
    }
    dependencies {
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin: $kotlin_version"
    }
}

apply plugin: 'kotlin'

dependencies {
    compile "org.jetbrains.kotlin:kotlin-stdlib: $kotlin_version"
}
```

~800KB

IDEA:

- automatically: ‘Configure Kotlin In Project’ option
- manually:
  - gradle: 14 lines(could be even less)
  - maven: didn’t fit :)

ECLIPSE:

- manually: install Kotlin plugin from marketplace

Yet another JVM language

Makes your life easier

```java
static String join(String sep, List<String> strings) {
    if (sep == null) throw new NullPointerException("sep == null");
    if (strings == null) throw new NullPointerException("strings == null");

    if (sep.length() < 2) throw new IllegalArgumentException("sep length < 2 " + sep);

    // ...
}
```

```kotlin
fun join(sep: String, strings: List<String>) : String {
    require(sep.length >= 2) { "sep length < 2 $sep" }

    // ...
}
```

```kotlin
class Person(val firstName: String, val lastName: String) {
    val name: String by lazy { "$firstName $lastName" }
}
```

```kotlin
import kotlin.LazyThreadSafetyMode.NONE

class Person(val firstName: String, val lastName: String) {
    val name: String by lazy(NONE) { "$firstName $lastName" }
}

// SYNCRONIZED, PUBLICATION, NONE
```

```
class Singleton private constructor() {
    var value: String? = null

    private object Holder {
        val INSTANCE = Singleton()
    }

    companion object {
        val instance: Singleton by lazy { Holder.INSTANCE }
    }
}

Singleton.instance.value = “changing value of Singleton”
```

```
object Singleton {
    var value: String? = null
}

Singleton.value = “changing value of Singleton”
```

```java
List<TrackDetail> items = Collections.emptyList();

public void addItems(List<TrackDetail> newItems) {
    items = newItems;
    notifyDataSetChange();
}

adapter.addItems(newItems);
```

```kotlin
var items: List<TrackDetail> by Delegates.observable(emptyList())
        { prop, old, new -> notifyDataSetChange() }

adapter.items = newItems
// Delegates.notNull()
// Delegates.vetoable()
```

```
interface BySatisfier {
    fun satisfyStrangeRequirements()
}

class Plafon(val gadget: String) : BySatisfier {
    override fun satisfyStrangeRequirements() = println("$gadget Plafon")
}

class Iphone(s) : BySatisfier by s


Iphone(Plafon("Iphone")).satisfyStrangeRequirements()
// output: Iphone Plafon
```

```java
interface BYSatisfier {
    void satisfyStrangeRequirements ();
}

class BYPlafon implements BYSatisfier {
    private final String gadget;
    public BYPlafon(String gadget) {
        this.gadget = gadget;
    }
    @Override
    public void satisfyStrangeRequirements () {
        System.out.println(gadget + " Plafon");
    }
}

class BYIphone implements BYSatisfier {
    private final BYSatisfier plafon;
    public BYIphone(BYSatisfier plafonchik) {
        this.plafon = plafonchik;
    }
    @Override
    public void satisfyStrangeRequirements () {
        plafon.satisfyStrangeRequirements() ;
    }
}
```

```java
interface BYSatisfier {
    void satisfyStrangeRequirements ();
}

class BYPlafon implements BYSatisfier {
    private final String gadget;
    public BYPlafon(String gadget) {
        this.gadget = gadget;
    }
    @Override
    public void satisfyStrangeRequirements () {
        System.out.println(gadget + " Plafon");
    }
}

class BYIphone implements BYSatisfier {
    private final BYSatisfier plafon;
    public BYIphone(BYSatisfier plafonchik) {
        this.plafon = plafonchik;
    }
    @Override
    public void satisfyStrangeRequirements () {
        plafon.satisfyStrangeRequirements() ;
    }
}
```

```kotlin
interface BYSatisfier {
    fun satisfyVeryStrangeTastes ()
}

class BYPlafon(gadget: String) : BYSatisfier{
    override fun satisfyVeryStrangeTastes () =
            println("$gadget Plafon")
}

class BYIphone(s: BelSatisfier) : BYSatisfier
        by s
```

```
SharedPreferences prefs = getSharedPreferences(APP_PREFS, Context.MODE_PRIVATE);
SharedPreferences.Editor editor = prefs.edit();
editor.putString("user", "Luke Skywalker");
editor.putInt("age", 19);
editor.apply();
```

```
inline fun SharedPreferences.edit(lambda: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    editor.lambda()
    editor.apply()
}

val prefs = getSharedPreferences(APP_PREFS, Context.MODE_PRIVATE)
prefs.edit {
    putString("user", "Luke Skywalker")
    putInt("age", 19)
}
```

```
inline fun SharedPreferences.edit(lambda: SharedPreferences.Editor.() -> Unit) {
    val editor = edit()
    editor.lambda()
    editor.apply()
}

fun <T> SharedPreferences.Editor.put(pair: Pair<String, T>) {
    when (pair.second) {
        is Boolean -> putBoolean(pair.first, pair.second as Boolean)
        is Int     -> putInt(pair.first, pair.second as Int)
        is String  -> putString(pair.first, pair.second as String)
        is Float   -> putFloat(pair.first, pair.second as Float)
        is Long    -> putLong(pair.first, pair.second as Long)
    }
}

val prefs = getSharedPreferences(APP_PREFS, Context.MODE_PRIVATE)
prefs.edit {
    put("user" to "Luke Skywalker")
    put("age" to 19)
    put("isDeathStarDestroyed" to true)
    remove("DeathStar")
}
```

AGAIN?

```
SharedPreferences.Editor editor = prefs.edit();
editor.putString("user", "Luke Skywalker");
editor.putInt("age", 19);
editor.apply();
```

```
prefs.edit {
    put("user" to "Luke Skywalker")
    put("age" to 19)
}
```

```
ContentValues values = ContentValues();//we've put (isReborned = true)
gotDatabase.beginTransaction();
try {
    gotDatabase.delete("characters", "full_name = ?", new String[] {"Ramsay Bolton"});
    gotDatabase.update("characters", values, "full_name = ?", new String[] {"John Snow"});
    gotDatabase.setTransactionSuccessful();
} finally {
    gotDatabase.endTransaction();
}
```

```
inline fun SQLiteDatabase.inTransaction(lambda: SQLiteDatabase.() -> Unit) {
    beginTransaction()
    try {
        lambda()
        setTransactionSuccessful()
    } finally {
        endTransaction()
    }
}

gotDatabase.inTransaction {
    delete("characters", "full_name = ?", arrayOf("Ramsay Bolton"))
    update("characters", values, "full_name = ?", arrayOf("John Snow"))
}
```

```
gotDatabase.beginTransaction();
try {
    gotDatabase.delete("characters", "full_name = ?", new String[] {"Ramsay Bolton"});
    gotDatabase.update("characters", values, "full_name = ?", new String[] {"John Snow"});
    gotDatabase.setTransactionSuccessful();
} finally {
    gotDatabase.endTransaction();
}
```

```
gotDatabase.inTransaction {
    delete("characters", "full_name = ?", arrayOf("Ramsay Bolton"))
    update("characters", values, "full_name = ?", arrayOf("John Snow"))
}
```

```
// We're inside activity
NotificationManager manager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
Notification notification = new Notification.Builder(context)
    .setContentTitle("Message")
    .setContentText("Wake up, Neo. The Matrix has you. Follow the White Rabbit.")
    .setContentIntent(enterTheMatrixIntent)
    .build();
manager.notify(NOTIFICATION_ID, notification);
```

```
inline fun Activity.showNotification(id: Int,
                                     lambda: Notification.Builder.() -> Unit) {
    val manager = getSystemService(NOTIFICATION_SERVICE) as NotificationManager
    val builder = Notification.Builder(context)
    builder.lambda()
    manager.notify(id, builder.build())
}

showNotification(NOTIFICATION_ID) {
    setContentTitle("Message")
    setContentText("Wake up, Neo. The Matrix has you. Follow the White Rabbit.")
    setContentIntent(enterTheMatrixIntent)
}
```

```
NotificationManager manager =
        (NotificationManager) context.getSystemService(NOTIFICATION_SERVICE);
Notification notification = new Notification.Builder(context)
        .setContentTitle("Message")
        .setContentText("Wake up, Neo. The Matrix has you. Follow the White Rabbit.")
        .setContentIntent(enterTheMatrixIntent)
        .build();
manager.notify(NOTIFICATION_ID, notification);
```

```
showNotification(NOTIFICATION_ID) {
    setContentTitle("Message")
    setContentText("Wake up, Neo. The Matrix has you. Follow the White Rabbit.")
    setContentIntent(enterTheMatrixIntent)
}
```

```java
List<Person> people = Collections.emptyList();
persons.add(new Person("John"));
persons.add(new Person("Jack"));
persons.add(new Person("Jill"));

List<String> names = people.stream().map(Person::getName).collect(Collectors.toList());
```

```kotlin
val people = listOf(Person("John"), Person("Jack"), Person("Jill"))
val names = people.map { Person::name }
```

```java
List<Person> persons = Collections.emptyList();
persons.add(new Person("John", Person.Sex.MALE));
persons.add(new Person("Jonathan", Person.Sex.MALE));
persons.add(new Person("Jill", Person.Sex.FEMALE));

Map<Person.Sex, List<String>> namesByGender = persons.stream()
        .collect(Collectors.groupingBy(Person::getGender,
                Collectors.mapping(Person::getName,
                        Collectors.toList())));
```

```kotlin
val persons = arrayOf(
            Person("John", Person.Sex.MALE),
            Person("Jonathan", Person.Sex.MALE),
            Person("Jill", Person.Sex.FEMALE))
val namesByGender = persons.groupBy { it.gender }.mapValues {
            it.value.map { it.name }
        }
```

```
/* Detect url type by last slug */
// url: https://bkug.by/2017/05/30/anons-bkug-4/    <- announce
// url: https://bkug.by/2017/03/25/otchet-o-bkug-3/ <- report
// url: https://bkug.by/2017/03/18/anons-bkug-3/    <- announce
/* The task is to print all the links to announces */
```

```
interface AnnounceUrlDetector {
    boolean isAnnounce(HttpUrl url);
}

class LastSlugIsAnnounce implements AnnounceUrlDetector {
    @Override
    public boolean isAnnounce(HttpUrl url) {
        final List<String> segments = url.pathSegments();
        final String lastSlug = segments.get(segments.size() - 1);

        if (lastSlug == null) return false;

        Pattern pattern = Pattern.compile("anons-bkug-\\d+");
        Matcher matcher = pattern.matcher(lastSlug);

        return matcher.matches();
    }
}
```

```java
interface AnnounceUrlDetector {
    boolean isAnnounce(HttpUrl url);
}
class LastSlugIsAnnounce implements AnnounceUrlDetector {
    @Override
    public boolean isAnnounce(HttpUrl url) {
        final List<String> segments = url.pathSegments();
        final String lastSlug = segments.get(segments.size() - 1);

        if (lastSlug == null) return false;

        Pattern pattern = Pattern.compile("anons-bkug-\\d+");
        Matcher matcher = pattern.matcher(lastSlug);

        return matcher.matches();
    }
}

final AnnounceUrlDetector isAnnounceDetector = new LastSlugIsAnnounce();
for (HttpUrl url : urls) {
    if (isAnnounceDetector.isAnnounce(url)) {
        System.out.println(url)
    }
}
```

```kotlin
interface UrlSlugDetector {
    fun detect(url: HttpUrl) : Boolean
}

class LastSlugDetector(val lambda: (String) -> Boolean) : UrlSlugDetector {
    override fun detect(url: HttpUrl): Boolean {
        val lastSlug = url.pathSegments().lastOrNull(String::isNotEmpty)
        return if (lastSlug == null) false else lambda(lastSlug)
    }
}

fun lastSlugIsAnnounce() = LastSlugDetector { Regex("anons-bkug-\d+") in it }
// in it = it.contains()

urls.filter { lastSlugIsAnnounce().detect(it) }.forEach(::println)
```

```
class LastSlugDetector implements UrlSlugDetector {


    private final Function<String, Boolean> lambda;
class LastSlugIsAnnounce implements AnnounceUrlDetector {

    @Override
    public LastSlugDetector(Function<String, Boolean> expr){
    public boolean isAnnounce(HttpUrl url) {
        lambda = expr;
        final List<String> segments = url.pathSegments();
    }
        final String lastSlug = segments

                .get(segments.size() - 1);
    @Override

    public boolean detect(HttpUrl url) {
        if (lastSlug == null) return false;
        final List<String> segments = url.pathSegments();

        final String lastSlug = segments
        Pattern pattern = Pattern.compile("anons-bkug-\d+");
                .get(segments.size() - 1);
        Matcher matcher = pattern.matcher(lastSlug);
        if (lastSlug == null) return false;

        return lambda.apply(lastSlug);
        return matcher.matches();
    }
    }
}
}


final LastSlugDetector isAnnounceDetector =
final AnnounceUrlDetector isAnnounceDetector = new
        new LastSlugDetector(lastSlug -> {
LastSlugIsAnnounce();
            Pattern pattern = Pattern
for (HttpUrl url : urls) {
                .compile("anons-bkug-\\d+");
    if (isAnnounceDetector.isAnnounce(url)) {
            Matcher matcher = pattern.matcher(lastSlug);
        System.out.println(url)

    }
            return matcher.matches();
}
});

urls.stream().filter(isAnnounceDetector::detect)

        .forEach(System.out::println);
```

```
class LastSlugDetector(val lambda: (String) -> Boolean) :
            UrlSlugDetector {


    override fun detect(url: HttpUrl): Boolean {
        val lastSlug = url.pathSegments()
                .lastOrNull(String::isNotEmpty)
        return if (lastSlug == null) false else lambda(lastSlug)
    }
}


fun lastSlugIsAnnounce() = LastSlugDetector {
        Regex("anons-bkug-\d+") in it
}


urls.filter { lastSlugIsAnnounce().detect(it) }
        .forEach(::println)
```

```java
Intent columbus = new Intent(SpainActivity.this, IndiaActivity.class);
intent.putExtra(KEY_ACTUAL_DESTINATION, AMERICA);
startActivity(columbus);
```

```kotlin
inline fun <reified T : Activity> Activity.navigate(lambda: Intent.() -> Unit) {
    val intent = Intent(this, T::class.java)
    intent.lambda()
    startActivity(intent)
}

/* Implying that we're inside SpainActivity */
navigate<IndiaActivity> { putExtra(KEY_ACTUAL_DESTINATION, AMERICA) }// first exped-n
navigate<WestIndiaActy> { putExtra(KEY_ACTUAL_DESTINATION, AMERICA) }// second exped-n
```

```
Intent columbus = new Intent(SpainActivity.this, IndiaActivity.class);
intent.putExtra(KEY_ACTUAL_DESTINATION, AMERICA);
startActivity(columbus);
```

```
navigate<IndiaActivity> { putExtra(KEY_ACTUAL_DESTINATION, AMERICA) }
```

```kotlin
sealed class Result<out T>
data class Success<out T>(val payload: T) : Result<T>()
data class Error<out T>(val message: String) : Result<T>()

data class UserResponse(val id: Int, val name: String, val surname: String, val subscriptionId: Int)

fun makeNetworkCall(type: Int) : Result<UserResponse> =when(type) {
    0 -> Success(UserResponse(112, "Andrey", "Breslav", 42))
    else -> Error("Something went wrong. Please, try again later.)"
}
fun <T> handleResponse(resp: Result<T>) {
    when(resp) { // Use concrete response as you wish; autocast below
        is Success -> {
            val (_, name, surname, _) = resp.payload as UserResponse
            fillNameViewWith("$name $surname")
        }
        is Error -> showError(resp.message)
    }
}

fun main(arr: Array<String>) {
    handleResponse(makeNetworkCall(Random().nextInt(2)))
}

// Output:
// random returns 0 -> Andrey Breslav fills the view
// random returns 1 -> Error view: Something went wrong. Please, try again later.
```

https://github.com/kittinunf/Result

```
public Observable<Pair<ISimModel, List<ISimModel>>>  selectedToOtherSimsPair (/* ... */)
```

```
typealias SimCardsPair = Pair<ISimModel, List<ISimModel>>>
fun selectedToOtherSimsPair (/* ... */): Observable<SimCardsPair>
```

```
public void onLanguagesReady (final List<Pair<String, String>>  languages,
                              final String defaultCode)
```

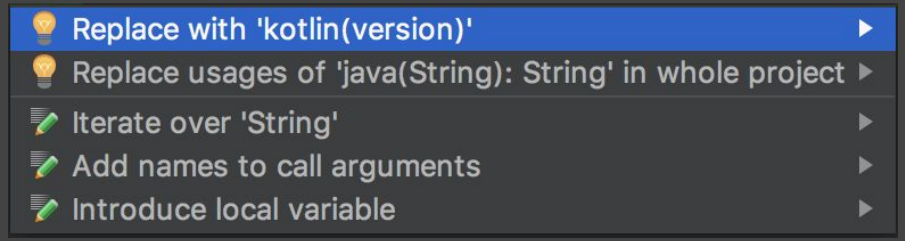
```
typealias CodesToLang = List<Pair<String, String>>
fun onLanguagesReady (languages: CodesToLang,  defaultCode: String)
```

```kotlin
@Deprecated("Use kotlin(version)", level = DeprecationLevel.ERROR)
fun java(version: String) : String {
    return "old $version"
}

java("1.6")

// console
Error:(52, 1) Kotlin: Using 'java(String): String' is an error. Use kotlin(version)
```

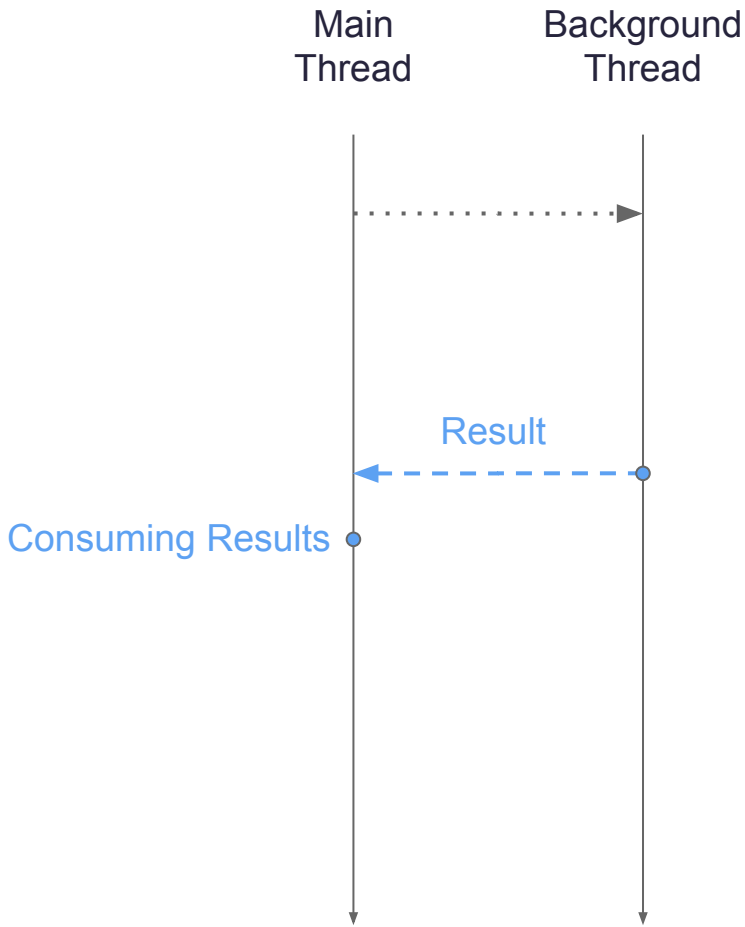
```kotlin
@Deprecated("Use kotlin(version)", level = DeprecationLevel.HIDDEN)
fun java(version: String) : String {
    return "java $version"
}

java("1.6") // no such method
```

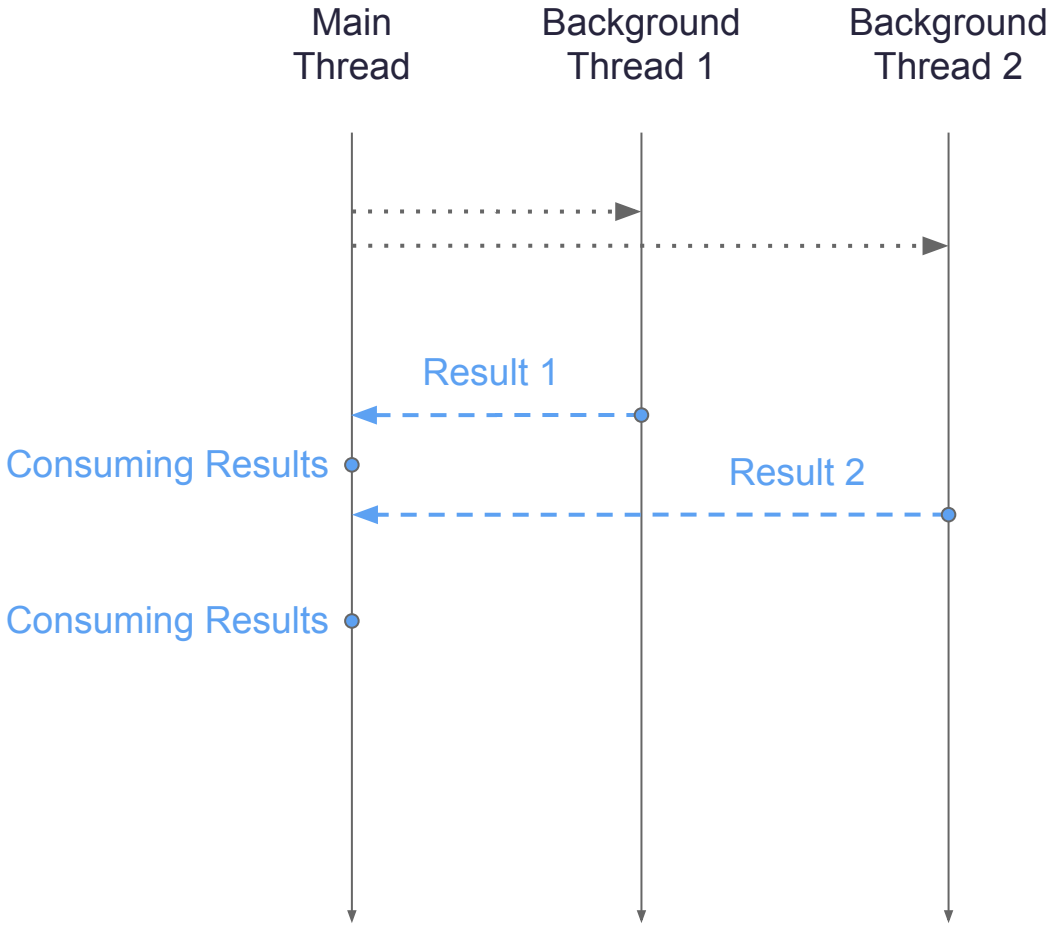
```
@Deprecated("Use kotlin(version)", ReplaceWith("kotlin(version)"), level =
        DeprecationLevel.WARNING)
fun java(version: String) : String {
    return "java $version"
}

fun kotlin(version: String) : String {
    return "kotlin $version"
}
```

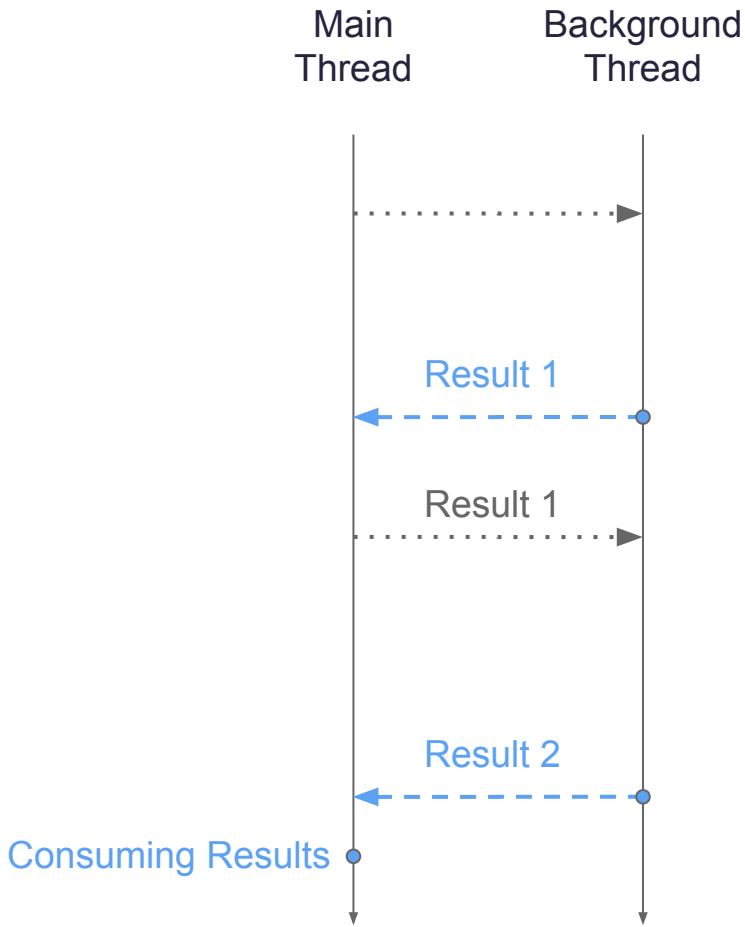
```
java("1.6")// don't forget to replace argument ;)
```



Coroutines


Main Thread
Background Thread
Result
Consuming Results


Main Thread
Background Thread 1
Background Thread 2
Result 1
Consuming Results
Result 2
Consuming Results


Main Thread
Background Thread
Result 1
Result 1
Result 2
Consuming Results

```
interface OnSenseWasFound {
   fun onSuccess(sense: Int = 42)
   fun onFail(message: String = "Life has no sense.")
}

fun findSenseOfLife(callback: OnSenseWasFound) { // your code here }
```

```
interface OnFeedbackReady {
    fun onAnswer(message: String = "Who cares?")
    fun onFail(message: String)
}

fun askHumanityFeedback(sense: Int, callback: OnFeedbackReady) { }


findSenseOfLife(object: OnSenseWasFound {
    override fun onSuccess(sense: Int) {
        askHumanityFeedback(sense, object: OnFeedbackReady {
            override fun onAnswer(message: String) {
                // code which handles result
            }
            override fun onFail(message: String) {
                tracker.reportError(message)
            }
        })
    }
    override fun onFail(message: String) {
        tracker.reportError(message)
    }
})
```

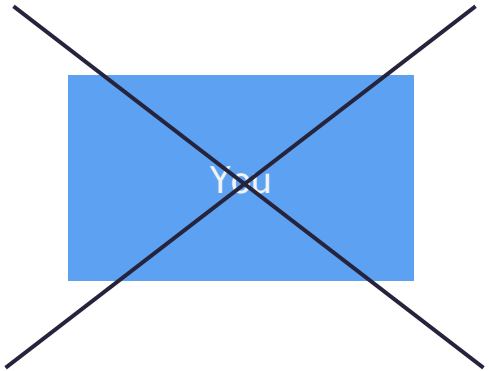
```
findSenseOfLife()
    .then(function(sense, failMessage))) {
        return askHumanityFeedback(sense)
    }.then(function(answer, failMessage)) {
        // code which handles result
    }
```

```
suspend fun findSenseOfLife() : Int { /* some code */ }
suspend fun askHumanityFeedback(sense: Int): String { /* some code */ }

launch(MainThread) {
    val sense = findSenseOfLife()
    val answer = askHumanityFeedback(sense)

    // code which handles result
}

// other code of your application
```

```
fun findSenseOfLife() : CompletableFuture<Int> = {
    CompletableFututre.supplyAsync {
        println("Searching...")
        42
    }
}

launch(MainThread) {
    val sense = findSenseOfLife().await()

    println("Answer is $sense")
}

println("Trying to find sense...")


// output: Trying to find sense…
// output: Searching…
// output: Answer is 42
```

```
val github = retrofit.create(GitHub::class.java)

launch(CommonPool)  {
   val contributors: List<Contributor> =
           github.contributors("JetBrains", "Kotlin")
                   .awaitSingle().take(10)

   for ((name, contributions) in contributors) {
       println("$name has $contributions contributions, other repos: ")

       val otherRepos = github.listRepos(name).awaitSingle()
               .map(Repo::name).joinToString(", ")

       println(otherRepos)
   }
}
println("We're fetching data...")

// output: We're fetching data…
// output: USER1 has 20 contributions, other repos:
// output: REPO1, REPO2, REPO3
// output: USER2 has 212 contributions, other repos:
// ...
// output: USER10, has 2 contributions, other repos:
```

Conclusion



Management

Team

- concise
- null safety
- 100% interop with Java (~99%) [means, you can use all your fav java libs]
- coroutines
- extension functions + lambdas to extend APIs like you want
- allows to make a step to higher level of abstraction (functional style)

REMOVES BOILERPLATE

- Latest version right now - 1.1.2 (JVM & Android & JS support)
- Popularity grows exponentially (by Github)
- Native support in 1.2.0 (C/C++ => IoT, iOS/macOS)
- In general, oriented to type-safe full-stack development

If you’re happy with Java there is ABSOLUTELY no need to migrate your codebase

Where to go next:

- http://kotlin.link - *tons of useful information about Kotlin*
- http://developer.android.com/kotlin/index.html - *Kotlin is occupying Android web site*
- kotlinlang.slack.com - *Official Kotlin's slack community (just google for free invite)*
- http://bkug.by - *Belarussian Kotlin User Group(🍺) <- subscribe for updates!*

Thanks!

Kotlin slack: @yev
twitter: @Eugene_Kovaliov

KOTLIN IN ACTION
BKUG Meetup #4

06/13/2017