



# HOW WE BUILD AND MIGRATED

our Spring Boot Applications to Kotlin

# WARNING

This meeting will contain live coding,  
so everything could (potentially) go wrong.

You will learn how to migrate a Spring Boot application  
from Java to Kotlin

Spoiler: It's not only the IntelliJ IDEA migration wizard....



# 5 Tips for moving



IMAGE FROM

[HTTPS://WWW.BROAVLOERISOLATIE.NL/](https://www.broavloerisolatie.nl/)



I'm gonna sue  
you for this!!!!



BIDNESS ETC



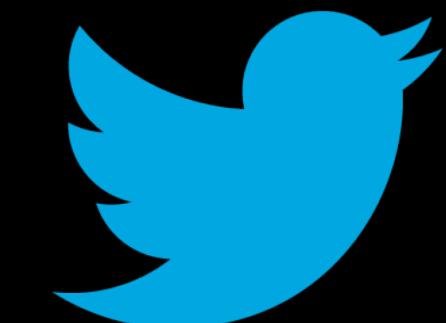






# Ko Turk

Developer



<https://github.com/KoTurk>

I'm telling my experiences

Kotlin ambasador

Not a Kotlin Champion

<https://twitter.com/KoTurk77>

# Agenda

- Different strategies
  - Class
  - Package
  - All
- It's not only the IntelliJ IDEA wizard











IMAGE FROM

[HTTPS://WWW.MYCHOICEHOMEZ.COM/](https://www.mychoicehomez.com/)



# Key takeaways:

- simplicity      -> **migrating is simple!**
- begin simple      -> **use data classes, do not start with logic**
- do safe calls      -> **with .let, .also and ?:**
- make your code readable      -> **with idiomatic Kotlin!**

# THANK YOU

Github: <https://github.com/KoTurk/>

Twitter: @KoTurk77



# ABOUT COROUTINES

Coroutines aren't a snap-in replacement for threads. The fundamental difference is in the kind of API you're using. The rule of thumb is this:

- blocking API -> use a thread pool
- non-blocking (async) API -> use coroutines

So, if you can get a hold of an asynchronous mail-sending API, then by all means use coroutines with it. But if you're stuck with a blocking API, coroutines won't bring you much value. They can make it a bit more convenient to transfer a blocking operation out of the UI thread, but the mechanics will be the same with or without coroutines.