# Reaktive Server mit Spring WebFlux und Kotlin Coroutines



Jochen Kraushaar


@KraushaarJochen


jochen.kraushaar@bridging-it.de


# Reaktive Server

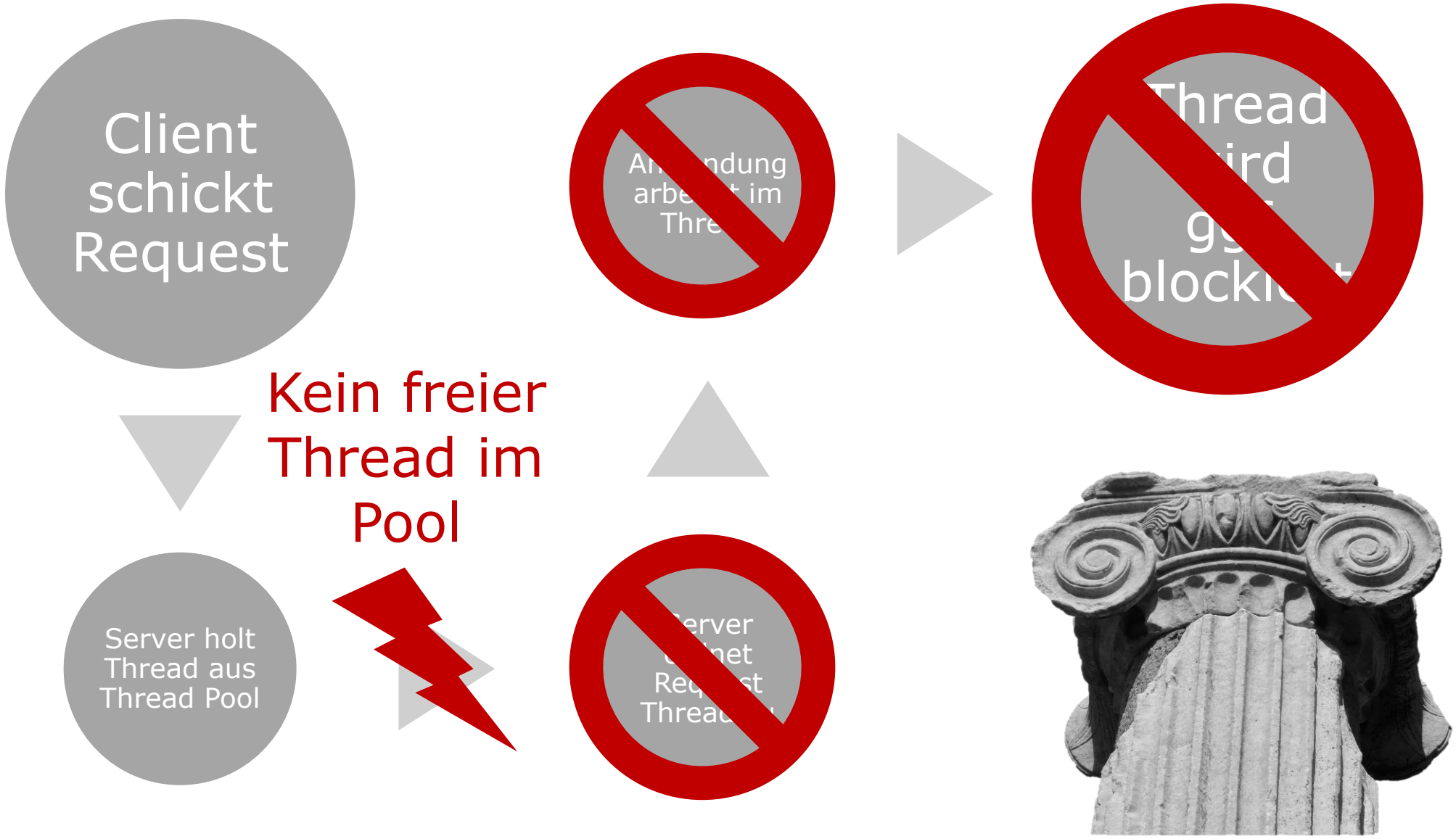
Client schickt Request

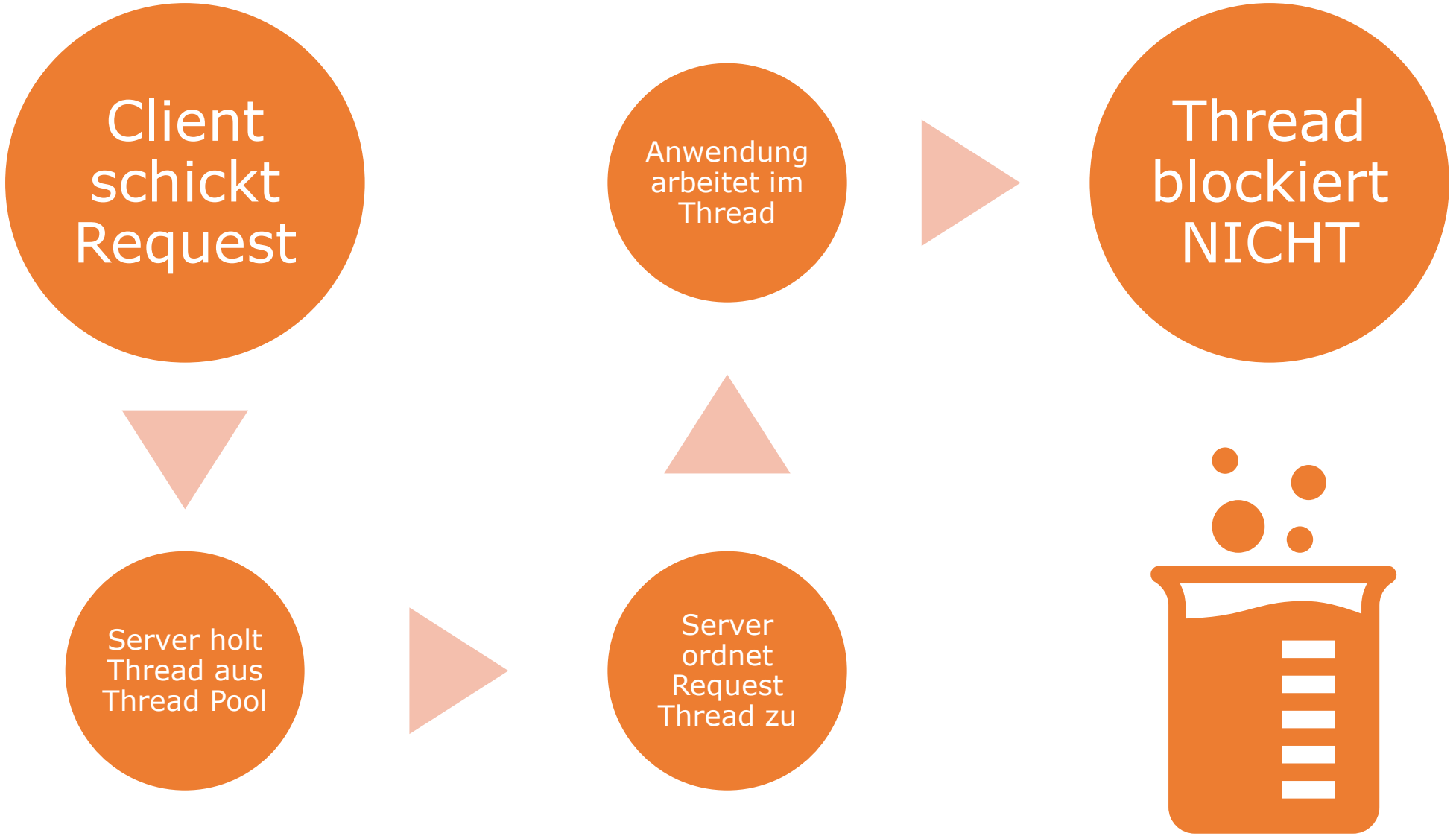
Server holt Thread aus Thread Pool

Server ordnet Request Thread zu

Anwendung arbeitet im Thread

Thread wird ggf. blockiert


Client schickt Request
Kein freier Thread im Pool
Server holt Thread aus Thread Pool


Client schickt Request
Server holt Thread aus Thread Pool
Server ordnet Request Thread zu
Anwendung arbeitet im Thread
Thread blockiert NICHT

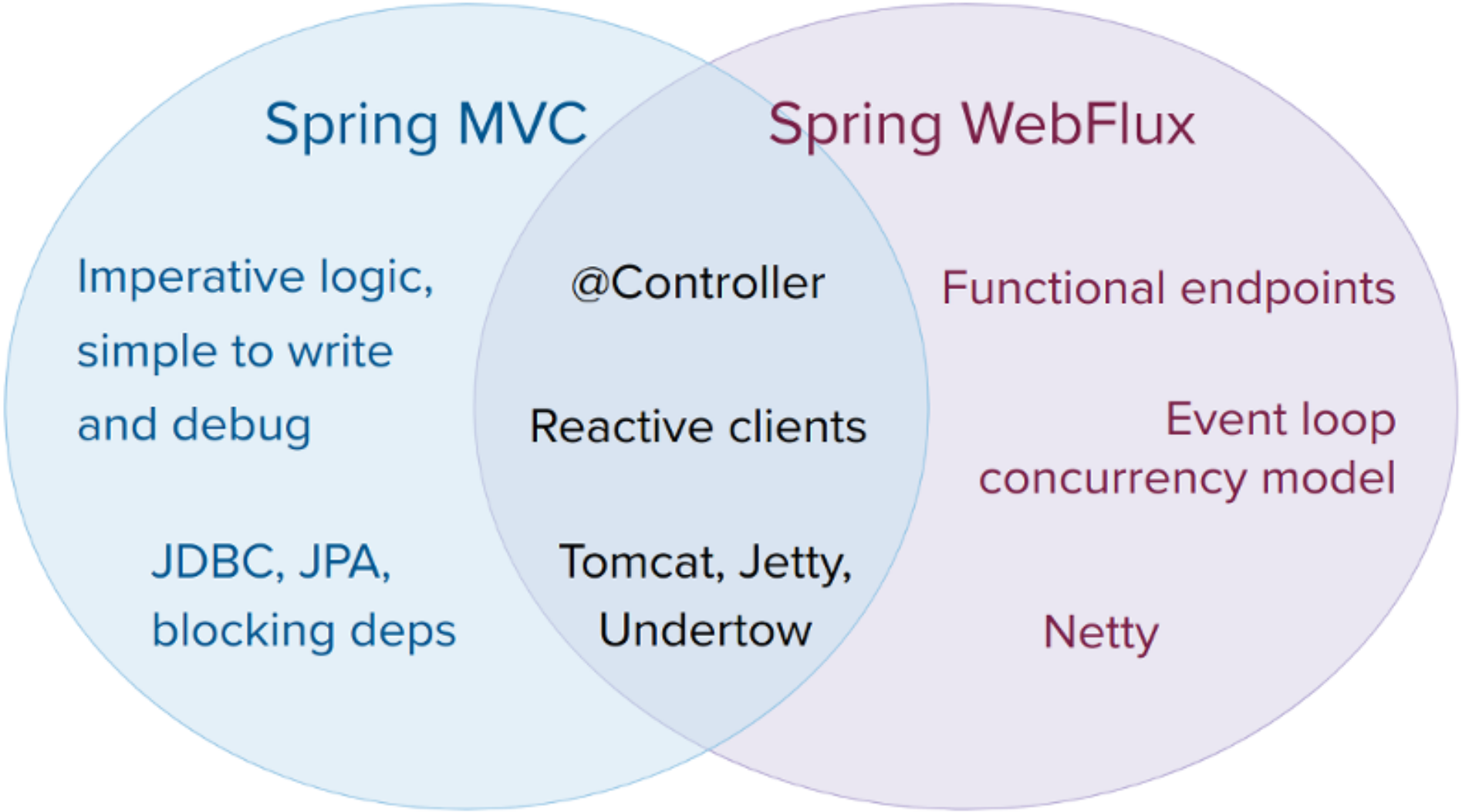
# Reaktiv = Non-Blocking

(mit Backpressure)

# Reaktive Streams

RxJava
Project Reactor

# Spring WebFlux




Quelle: https://docs.spring.io/spring/docs/5.2.0.RELEASE/spring-framework-reference/web-reactive.html

# Mono

Project Reactor

0..1 Elemente

„Optional auf Steroide“

```
@GetMapping("/{id}")
fun findAvailableProduct(@PathVariable id: Int): Mono<AvailableProduct> =
    productRepository.getProductById(id)
            .filter { !it.deleted }
            .map { AvailableProduct(it) }
```

# Flux

Project Reactor
0..N Elemente
Reaktiver Stream

```
@GetMapping("/")
fun findAllAvailableProducts(): Flux<AvailableProduct> =
    productRepository.getAllProducts()
            .filter { !it.deleted }
            .map { AvailableProduct(it) }
```

# Achtung!

- Deklarativ statt Imperativ
- Exceptions als Rückgabe
- Reaktiver Stack
- Kein JDBC

# R2DBC

Spezifikation

Version 1.0.0.RC1

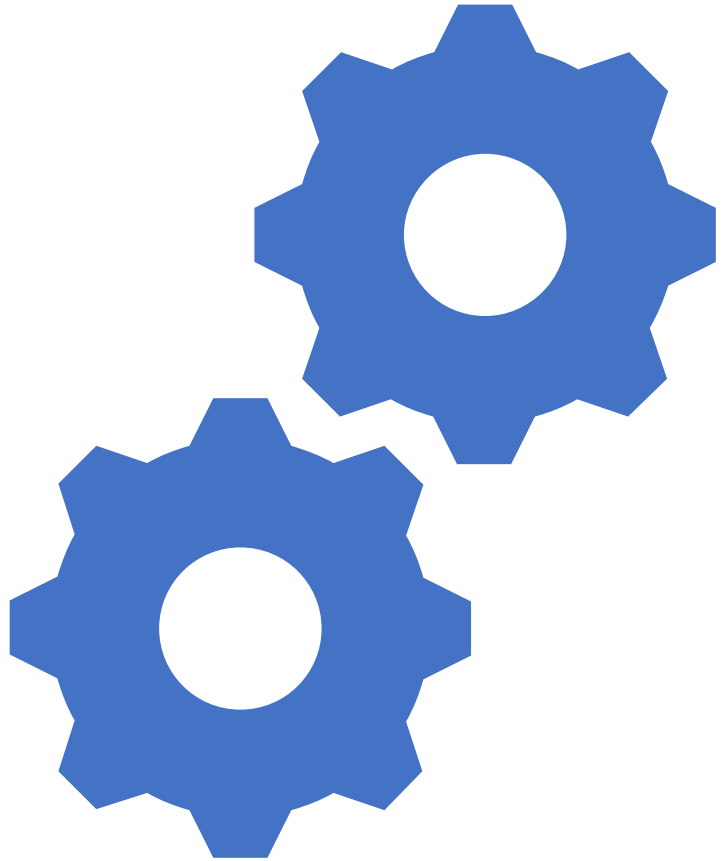
H2

PostgreSQL

Microsoft SQL Server

MySQL

# Kotlin Coroutines



# Warum?

Imperativ

„Leichtgewichtige Threads“

Suspending Functions

# Coroutine Scope

```
fun main() = runBlocking {
    repeat(100_000) { // launch a lot of coroutines
        launch {
            delay(1000L)
            print(".")
        }
    }
}
```

Quelle: https://kotlinlang.org/docs/reference/coroutines/basics.html, Apache 2 License

suspend

```kotlin
suspend fun doSomethingUsefulOne(): Int {
    delay(1000L) // pretend we are doing something useful here
    return 13
}

suspend fun doSomethingUsefulTwo(): Int {
    delay(1000L) // pretend we are doing something useful here, too
    return 29
}
```

Quelle: https://kotlinlang.org/docs/reference/coroutines/composing-suspending-functions.html, Apache 2 License

# async / await

```kotlin
fun main() = runBlocking<Unit> {
    val time = measureTimeMillis {
        val one = async { doSomethingUsefulOne() }
        val two = async { doSomethingUsefulTwo() }
        println("The answer is ${one.await() + two.await()}")
    }
    println("Completed in $time ms")
}
```

Quelle: https://kotlinlang.org/docs/reference/coroutines/composing-suspending-functions.html, Apache 2 License

# Mono vs. suspend

```
@GetMapping("/{id}")
fun findAvailableProduct(@PathVariable id: Int): Mono<AvailableProduct> =
    productRepository.getProductById(id)
            .filter { !it.deleted }
            .map { AvailableProduct(it) }
```

```
@GetMapping("/{id}")
suspend fun findAvailableProduct(@PathVariable id: Int): AvailableProduct?
{
    val product = productRepository.getProductById(id) ?: return null
    return AvailableProduct(product)
}
```

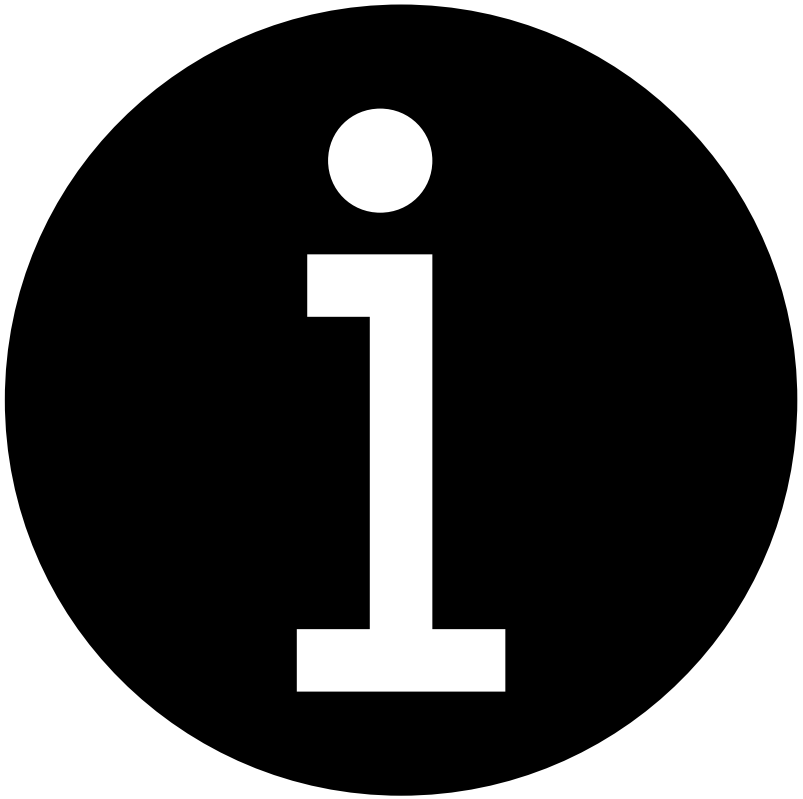
# Flux vs. Flow vs. suspend

```
@GetMapping("/")
fun findAllAvailableProducts(): Flux<AvailableProduct> =
    productRepository.getAllProducts()
            .filter { !it.deleted }
            .map { AvailableProduct(it) }
```

```
@GetMapping("/")
fun findAllAvailableProducts(): Flow<AvailableProduct> =
    productRepository.getAllProducts()
            .filter { !it.deleted }
            .map { AvailableProduct(it) }
```

```kotlin
@GetMapping("/")
suspend fun findAllAvailableProducts(): List<AvailableProduct> {
    val products = productRepository.getAllProducts().toList()
    return products.filter { !it.deleted }.map { AvailableProduct(it) }
}
```

# Fazit & Mehr



# Brauche ich Reaktive Server?

# Gut für moderne Cloud-Architekturen

# Aber…

Keine Blocking APIs

Anderes Programmiermodell

Kotlin Coroutines nicht erforderlich, aber empfehlenswert

# Wie gut funktioniert Kotlin mit Spring?

# Sehr gut

„First-Class Support“
Spring Initializr
Spring Boot KoFu
Kotlin Coroutines

# Danke!



Jochen Kraushaar


@KraushaarJochen


jochen.kraushaar@bridging-it.de


## Quellen

- https://github.com/ReactiveX/RxJava
- https://projectreactor.io/
- https://r2dbc.io/
- https://docs.spring.io/spring/docs/current/spring-framework-reference/web-reactive.html
- https://www.baeldung.com/spring-boot-kotlin-coroutines
- https://spring.io/blog/2019/04/12/going-reactive-with-spring-coroutines-and-kotlin-flow
- https://kotlinlang.org/docs/reference/coroutines/

# BACKLOG

# Reaktiv zum Client

HTTP (Websockets, Server-Side Events)

HTTP/2

TCP

UDP