Live coding server as a function with http4k

@dmitrykandalov github.com/dkandalov youtube.com/dkandalov

Your Server as a Function

Marius Eriksen

Twitter Inc.

marius@twitter.com

Abstract

Building server software in a large-scale setting, where systems exhibit a high degree of concurrency and environmental variability, is a challenging task to even the most experienced programmer. Efficiency, safety, and robustness are paramount—goals which have traditionally conflicted with modularity, reusability, and flexibility.

We describe three abstractions which combine to present a powerful programming model for building safe, modular, and efficient server software: Composable *futures* are used to relate concurrent, asynchronous actions; *services* and *filters* are specialized functions used for the modular composition of our complex server software.

Finally, we discuss our experiences using these abstractions and techniques throughout Twitter's serving infrastructure.

Categories and Subject Descriptors D.1.1 [*Programming techniques*]: Applicative (Functional) Programming; D.1.3 [*Programming techniques*]: Concurrent Programming; D.1.3 [*Program-*

Services Systems boundaries are represented by asynchronous functions called *services*. They provide a symmetric and uniform API: the same abstraction represents both clients and servers.

Filters Application-agnostic concerns (e.g. timeouts, retries, authentication) are encapsulated by *filters* which compose to build services from multiple independent modules.

Server operations (e.g. acting on an incoming RPC or a timeout) are defined in a declarative fashion, relating the results of the (possibly many) subsequent sub-operations through the use of future combinators. Operations are phrased as *value transformations*, encouraging the use of immutable data structures and, we believe, enhancing correctness through simplicity of reasoning.

Operations describe *what* is computed; execution is handled separately. This frees the programmer from attending to the minutiae of setting up threads, ensuring pools and queues are sized cor-

http4k

Documentation Changelog Support/Training FAQ Toolbox

The Functional toolkit for Kotlin HTTP applications

http4k provides a simple and uniform way to serve, consume, and test HTTP services.

Quick Start

HTTP as a Function

```
val app = { request: Request ->
    Response(OK)
        .body("Hello, ${request.query("name")}!")
}

app.asServer(Jetty(9000)).start()
```

```
val client = ApacheClient()

val request = Request(GET, "http://localhost:9000")
                .query("name", "John Doe")

val response: Response = client(request)
```

http4k / http4k

Public

Sponsor | Watch 30 | Unstar 1.9k | Fork 178

Code | Issues 27 | Pull requests 3 | Actions | Projects | Security | Insights

master | 13 branches | 100 tags | Go to file | Add file | Code

daviddenton fix #679 - approval actual files are deleted w... a04ff9a 2 days ago **5,825** commits

.github	Release 4.17.2.0	5 days ago
gradle/wrapper	upgrades	6 months ago
http4k-aws	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-bom	upgrades and convert build to Kotlin (#663)	3 months ago
http4k-client	deprecate rename AsyncHttpClient to AsyncHttpHandler	2 months ago
http4k-cloudevents	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-cloudnative	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-contract	Servers in contracts (#674)	5 days ago
http4k-core	Test is nicer	5 days ago
http4k-format	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-graphql	fix graphql	2 months ago
http4k-incubator	upgrades and convert build to Kotlin (#663)	3 months ago
http4k-jsonrpc	upgrades and convert build to Kotlin (#663)	3 months ago

About

The Functional toolkit for Kotlin HTTP applications. http4k provides a simple and uniform way to serve, consume, and test HTTP services.

http4k.org

kotlin | http | tdd | http-client | immutability | http-server | typesafe | testability | http4k

Readme

Apache-2.0 License

Code of conduct

Releases 99

4.17.2.0 Latest
5 days ago

+ 98 releases

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

In other projects

Tic-tac-toe

From Wikipedia, the free encyclopedia

"Tic Tac Toe" and "Noughts and crosses" redirect here. For other uses, see Tic Tac Toe (disambiguation) and Noughts and crosses (disambiguation).

Tic-tac-toe (American English), **noughts and crosses** (British English), or **Xs and Os** is a paper-and-pencil game for two players, *X* and *O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.

The following example game is won by the first player, X:

Players soon discover that the best play from both parties leads to a draw. Hence, tic-tac-toe is most often played by young children, who often have not yet discovered the optimal strategy.

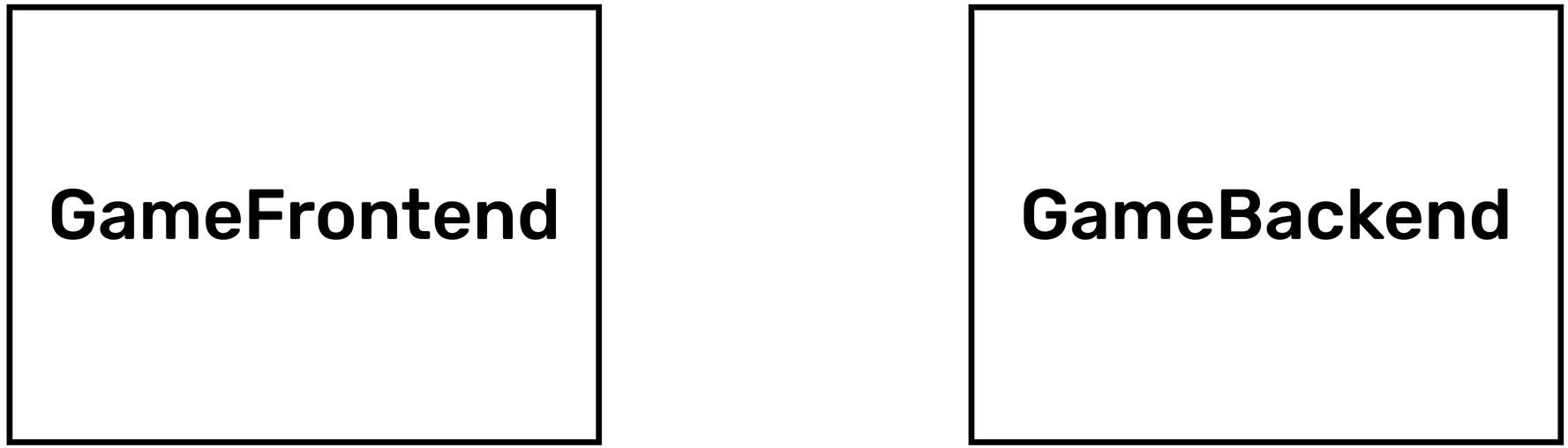
Because of the simplicity of tic-tac-toe, it is often used as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees. It is straightforward to write a computer program to play tic-tac-toe perfectly or to enumerate the 765 essentially different positions (the state space complexity) or the 26,830 possible games up to rotations and reflections (the game tree complexity) on this space.[1]

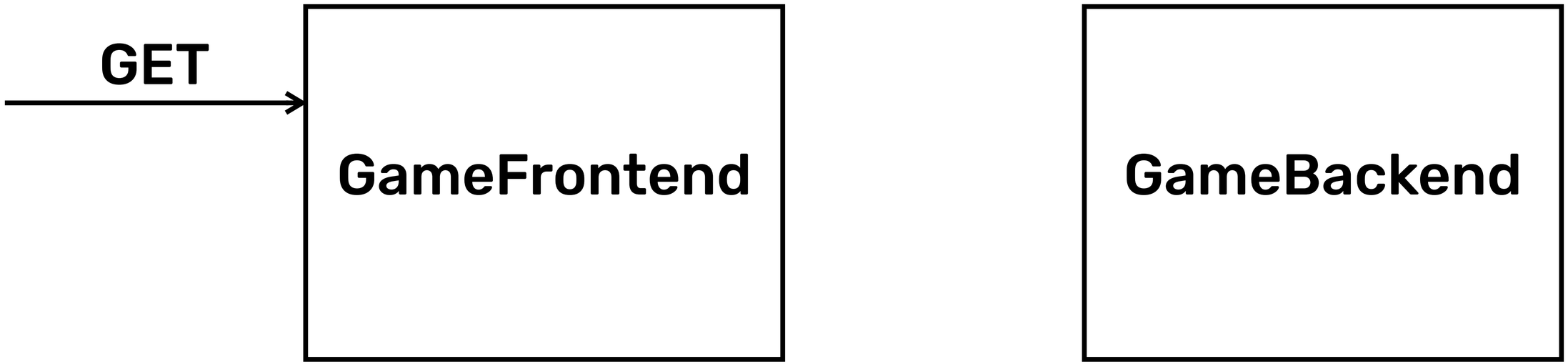
The game can be generalized to an m,n,k-game in which two players alternate placing stones of their own color on an *m*×*n* board, with the goal of getting *k* of their own color in a

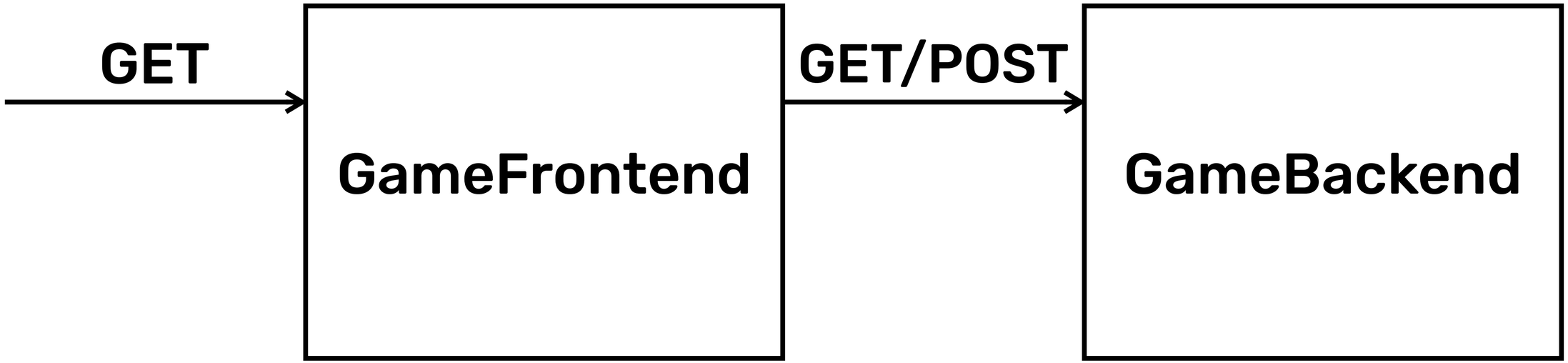
Tic-tac-toe

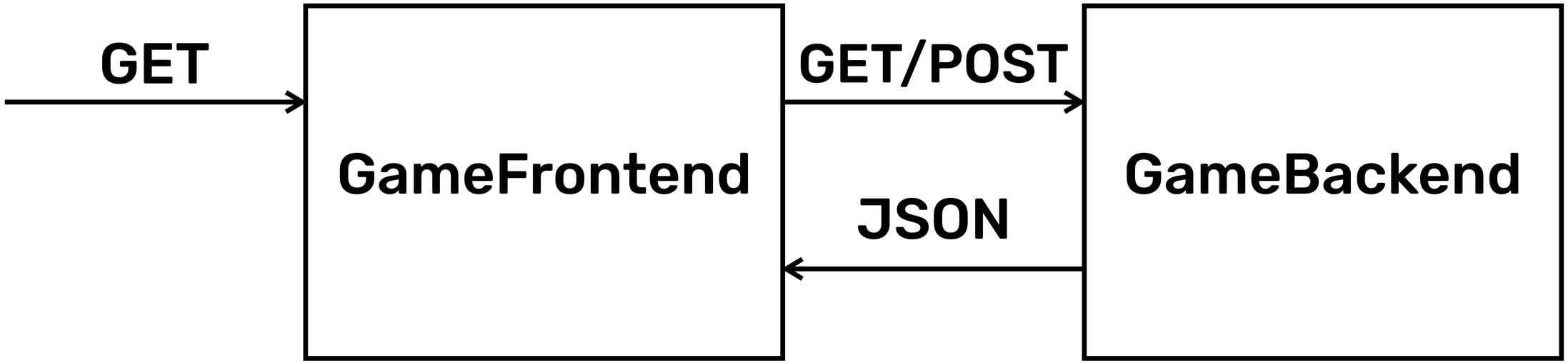
A completed game of Tic-tac-toe

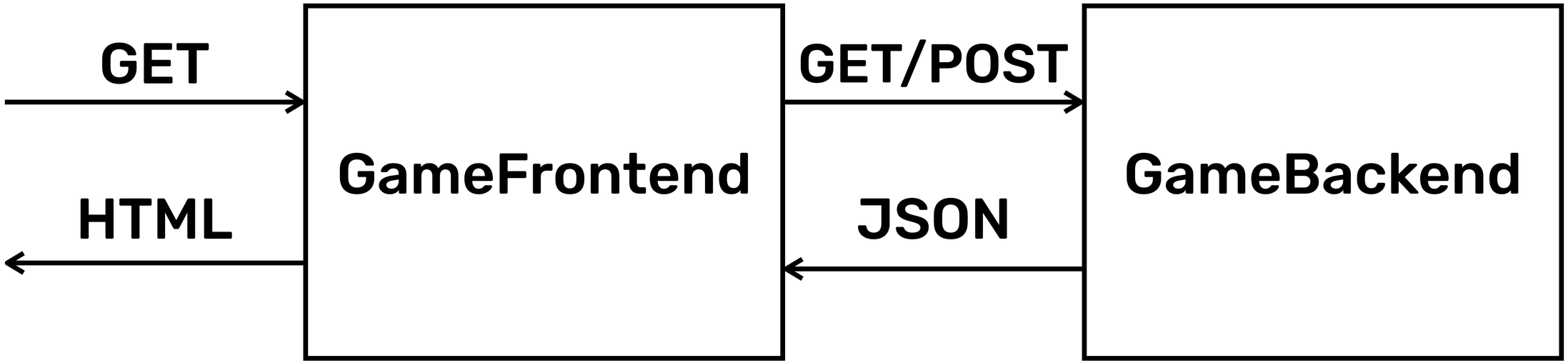
Genre(s)	Paper-and-pencil game
Players	2
Setup time	Minimal
Playing time	~1 minute
Random chance	None
Skill(s) required	Strategy, tactics, observation
Synonym(s)	Noughts and crosses Xs and Os

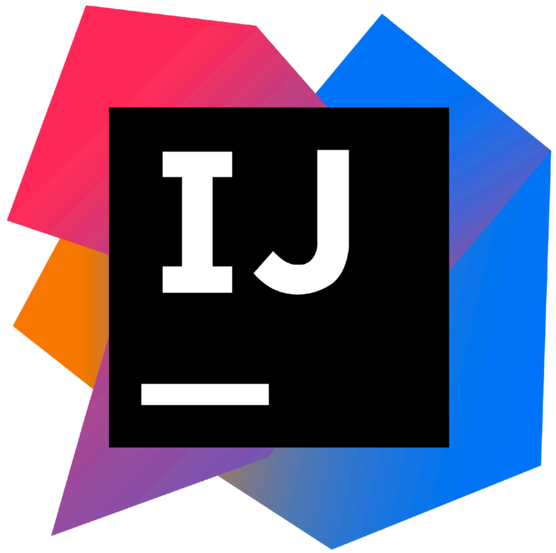

GameFrontend
GameBackend

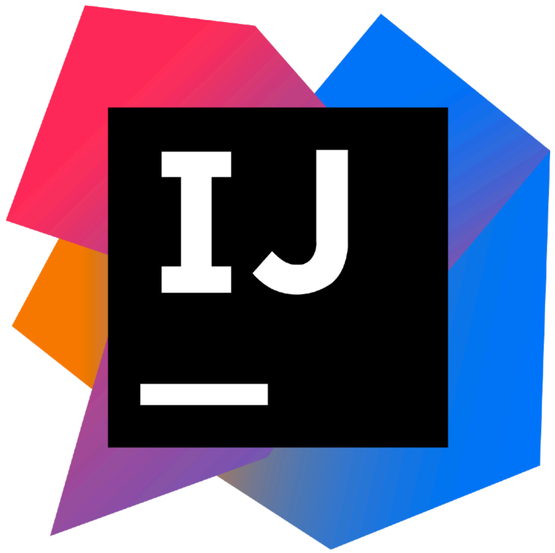

GET
GameFrontend
GameBackend


GET
GameFrontend
GET/POST
GameBackend


GET
GameFrontend
GET/POST
JSON
GameBackend


GET
HTML
GameFrontend
GET/POST
JSON
GameBackend


IJ


IJ

There are more http4k things…

- more clients/servers

- more clients/servers
- more JSON/XML libs

- more clients/servers
- more JSON/XML libs
- more template engines

- more clients/servers
- more JSON/XML libs
- more template engines
- servless platforms

- approval tests

- approval tests
- contract tests

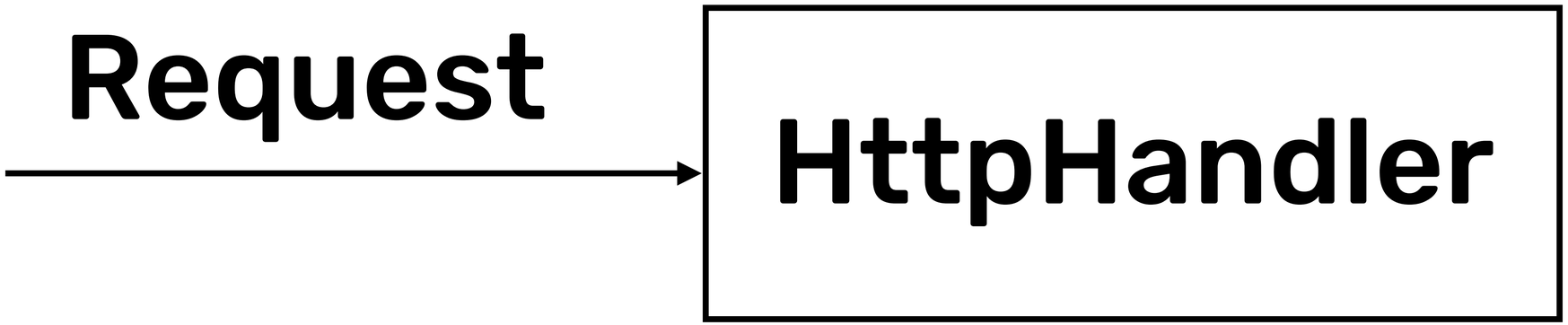
- approval tests
- contract tests
- chaos testing

- approval tests
- contract tests
- chaos testing
- record/replay traffic

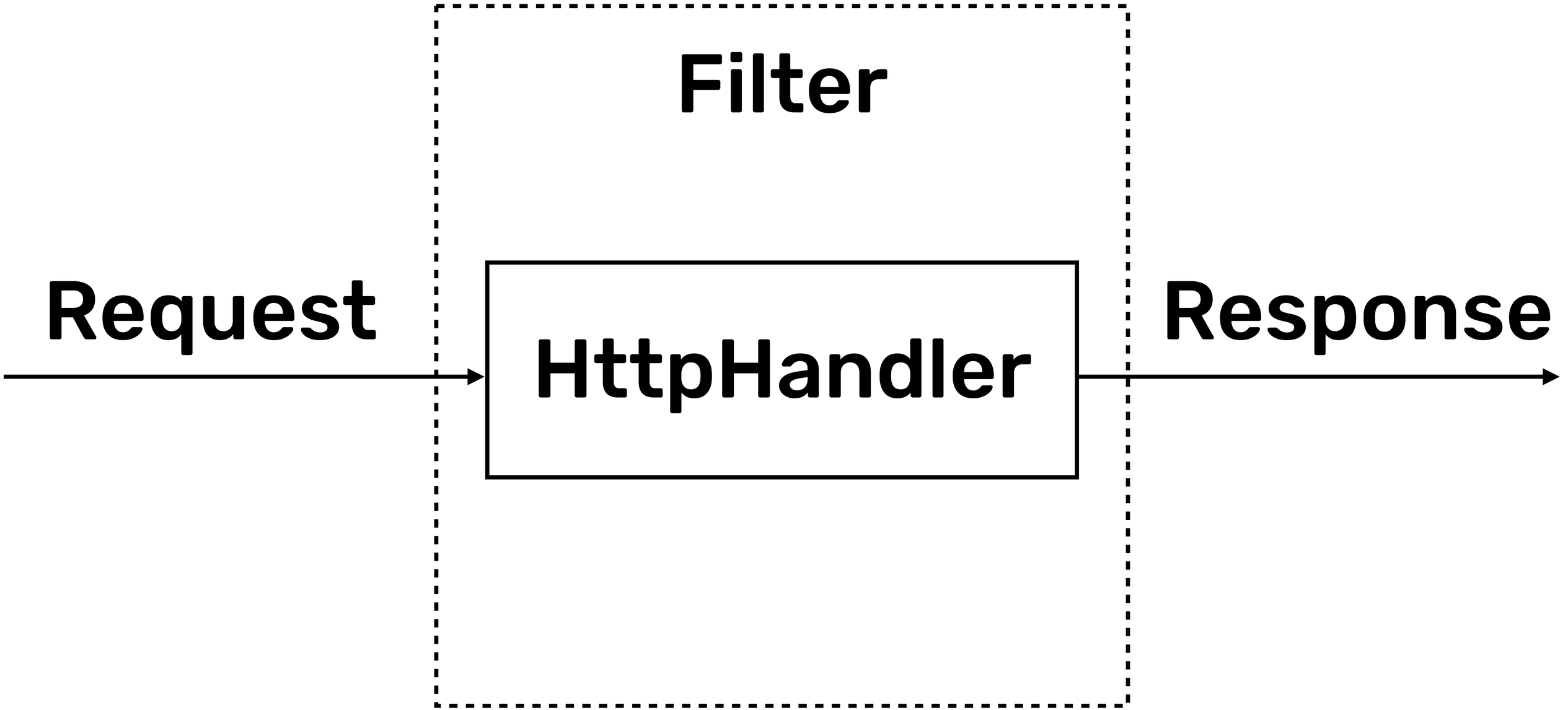
One thing to remember…

Client and server have the same type signature!

HttpHandler


Request
HttpHandler


Request
HttpHandler
Response


Filter
Request
HttpHandler
Response

Next steps…

dkandalov / tictactoe4k

Public

Unwatch 1 | Unstar 7 | Fork 2

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 2 branches | 0 tags | Go to file | Add file | Code

dkandalov updated dependencies — 7a6e6ea 3 days ago — **44** commits

.github/workflows	added git workflow	15 months ago
.live-plugins	added console filter for project-specific plugin	3 months ago
gradle/wrapper	updated dependencies	6 months ago
slides	updated slides	10 months ago
src/test	use http4k-format-moshi	5 months ago
.gitignore	initial commit	2 years ago
build.gradle.kts	updated dependencies	3 days ago
gradlew	initial commit	2 years ago
gradlew.bat	initial commit	2 years ago
readme.md	updated readme	10 months ago
settings.gradle.kts	kotlin 1.4-M1; gradle 6.3	2 years ago

readme.md

About

Source code and slides for the tictactoe4k talk

www.youtube.com/watch?v=fvvn-afo-...

Readme

Languages

Kotlin 54.9% Groovy 42.1%

Handlebars 3.0%

http4k / http4k-by-example

Public

Sponsor | Watch 2 | Star 38 | Fork 6

Code | Issues 1 | Pull requests | Actions | Projects | Wiki | Security | Insights

master | 1 branch | 0 tags | Go to file | Add file | Code

daviddenton fix invalidating all cookies — 71ba86d 21 days ago — 157 commits

gradle/wrapper	upgrade http4k, fix deprecations	11 months ago
src	fix invalidating all cookies	21 days ago
.gitignore	added stuff to gitignore	2 years ago
.travis.yml	introduce travis	3 years ago
README.md	readme	2 years ago
build.gradle	update things..	8 months ago
gradle.properties	update things..	8 months ago
gradlew	upgrade gradle, remove cobertura	17 months ago
gradlew.bat	upgrade gradle, remove cobertura	17 months ago
settings.gradle	introduce shippable	3 years ago
shippable.yml	introduce shippable	3 years ago

README.md

About

Complete TDD'd example http4k application showcasing a lot of the http4k features for building apps

Readme

Releases

No releases published

Sponsor this project

http4k http4k

Sponsor

Learn more about GitHub Sponsors

Packages

No packages published

http4k / http4k

Public

Sponsor | Watch 30 | Unstar 1.9k | Fork 178

Code | Issues 27 | Pull requests 3 | Actions | Projects | Security | Insights

master | 13 branches | 100 tags | Go to file | Add file | Code

daviddenton fix #679 - approval actual files are deleted w... a04ff9a 2 days ago **5,825** commits

.github	Release 4.17.2.0	5 days ago
gradle/wrapper	upgrades	6 months ago
http4k-aws	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-bom	upgrades and convert build to Kotlin (#663)	3 months ago
http4k-client	deprecate rename AsyncHttpClient to AsyncHttpHandler	2 months ago
http4k-cloudevents	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-cloudnative	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-contract	Servers in contracts (#674)	5 days ago
http4k-core	Test is nicer	5 days ago
http4k-format	Upgrade to Kotlin 1.6 and associated changes (#675)	15 days ago
http4k-graphql	fix graphql	2 months ago
http4k-incubator	upgrades and convert build to Kotlin (#663)	3 months ago
http4k-jsonrpc	upgrades and convert build to Kotlin (#663)	3 months ago

About

The Functional toolkit for Kotlin HTTP applications. http4k provides a simple and uniform way to serve, consume, and test HTTP services.

http4k.org

kotlin | http | tdd | http-client | immutability | http-server | typesafe | testability | http4k

Readme

Apache-2.0 License

Code of conduct

Releases 99

4.17.2.0 Latest
5 days ago

+ 98 releases

http4k / http4k Public

Sponsor | Watch 30 | Unstar 1.9k | Fork 178

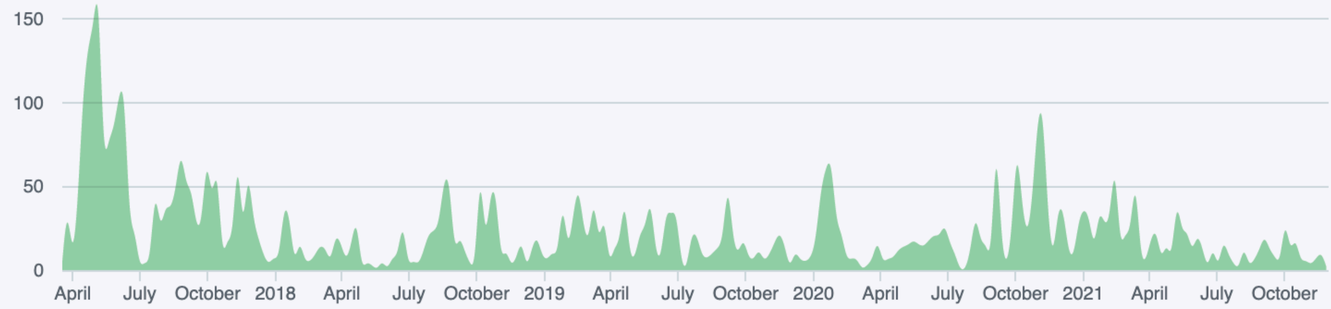
Code | Issues 27 | Pull requests 3 | Actions | Projects | Security | Insights

- Pulse
- Contributors
- Community
- Commits
- Code frequency
- Dependency graph
- Network
- Forks

Mar 19, 2017 – Dec 1, 2021

Contributions: Commits

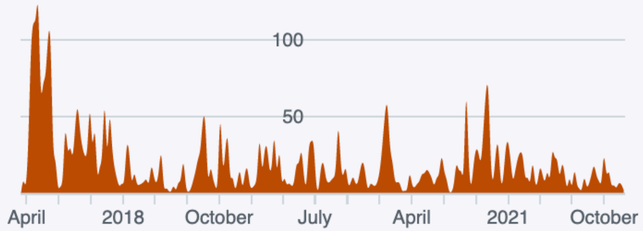
Contributions to master, excluding merge commits and bot accounts



daviddenton

#1

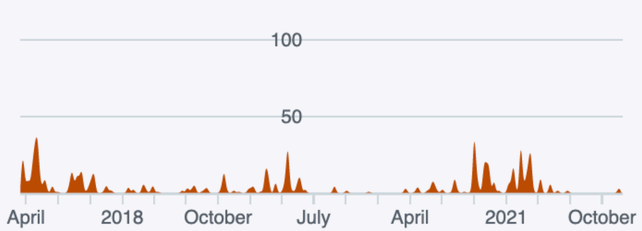
4,682 commits 294,233 ++ 231,024 --

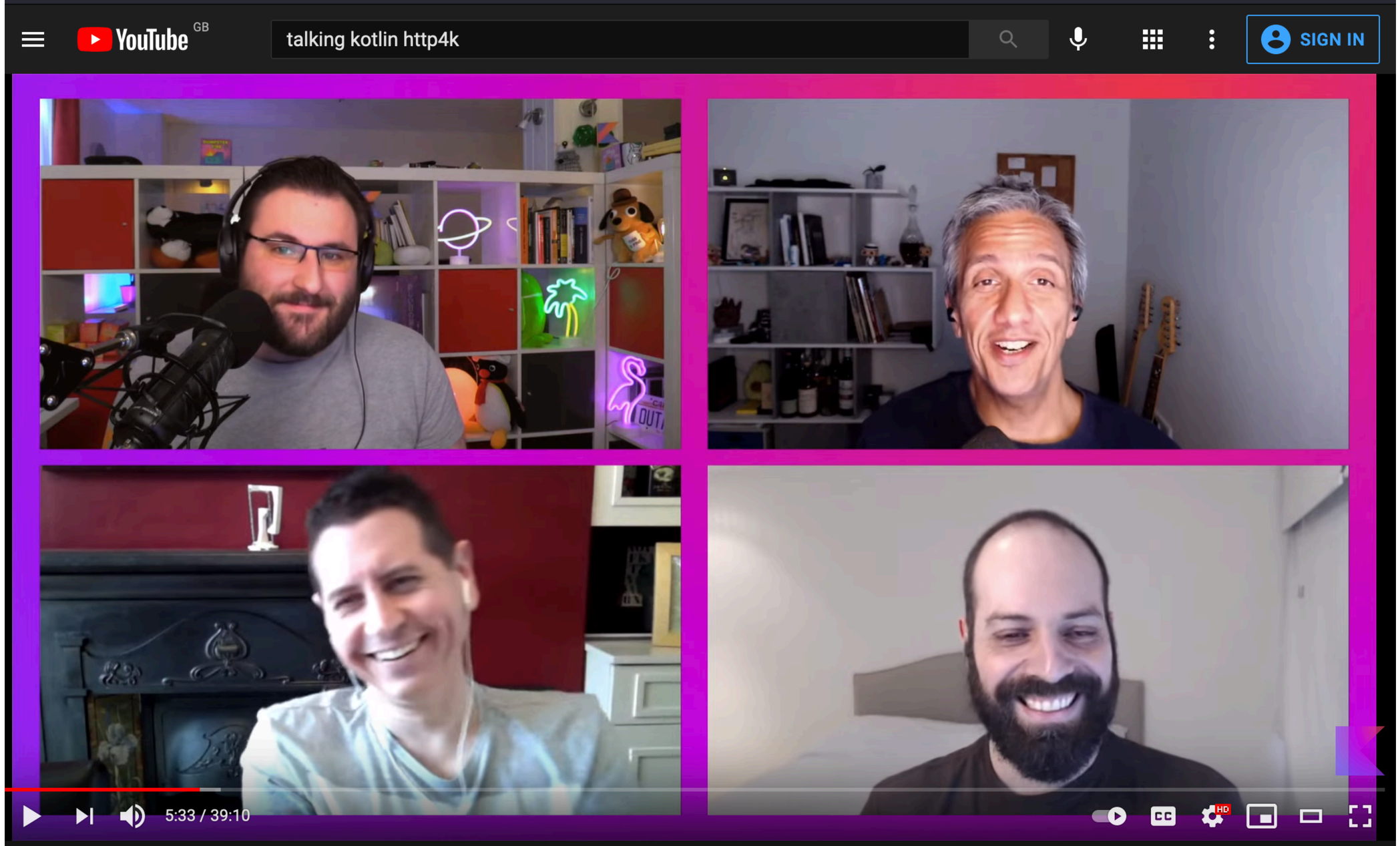


s4nchez

#2

861 commits 48,883 ++ 37,311 --





#http4k #serverless #kotlin

Talking Kotlin #99 HTTP as a Function With http4k

3,224 views • Jun 22, 2021

62 0 SHARE SAVE ...

Kotlin 1.5 Online Event

Kotlin by JetBrains

16K views • Streamed 1 month ago

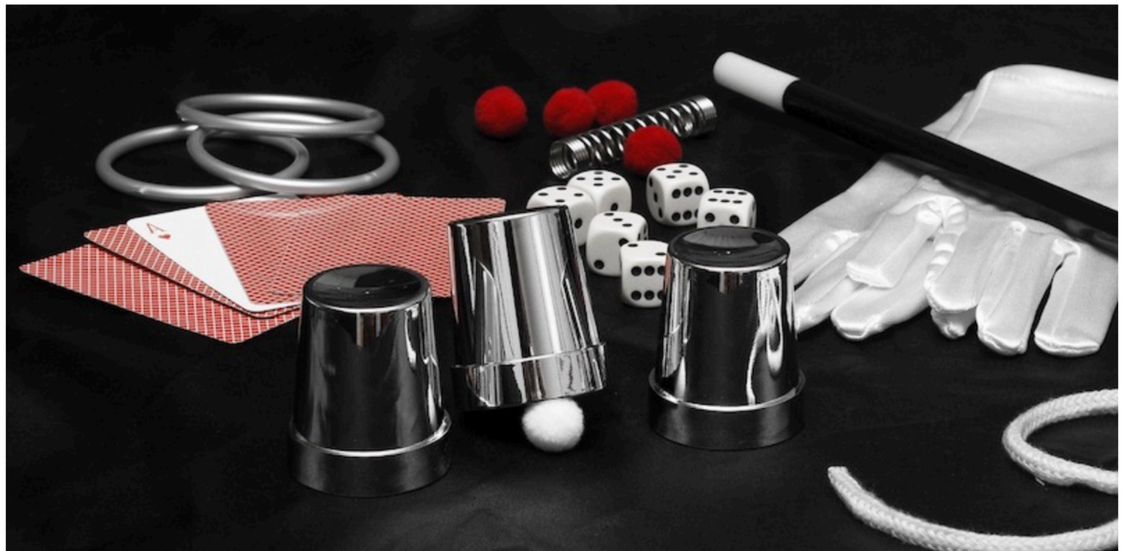
Refactoring Kotlin type-signatures for fun and profit

Posted on June 29, 2021

In which I derive the core type signatures of http4k from an unexpected source, by using refactoring and FP techniques more commonly associated with coding.

TL;DR

> *"You can apply simple refactoring and functional techniques such as substitution and currying to refine function signatures. Free of any implementation detail, defining system concepts in this way can be a refreshingly powerful technique."*



YouTube GB

test driven apps with http4k

Next On Track

Ivan Sanchez & David Denton

HTTP4K

Writing Test Driven Apps with http4k

YouTube GB

http4k kotlinconf

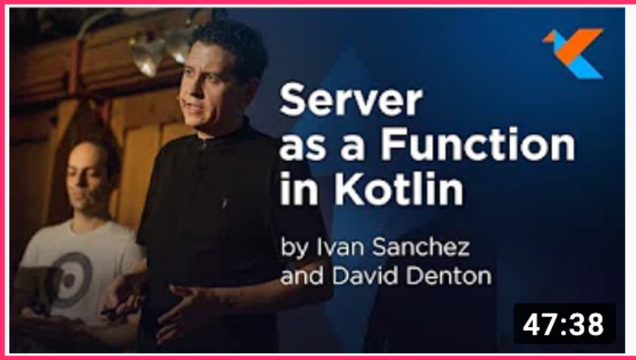
Home

Trending

Subscriptions

Library

FILTER



KotlinConf 2018 - Server as a Function in Kotlin by Ivan Sanchez & David Denton

JetBrainsTV • 2.1K views • 1 year ago

Recording brought to you by American Express https://americanexpress.io/kotlin-jobs In this talk, you'll learn about how we ...

KotlinConf 2019: Asynchronous Data Streams with Kotlin Flow by Roman Elizarov

JetBrainsTV • 15K views • 2 months ago

Recording brought to you by American Express. https://americanexpress.io/kotlin-jobs Kotlin Flow is a declarative mechanism ...

KotlinConf 2019: What's New in Java 19: The end of Kotlin? by Jake Wharton

JetBrainsTV • 111K views • 2 months ago

Recording brought to you by American Express. https://americanexpress.io/kotlin-jobs Kotlin's introduction was a breath of ...

http4k

Documentation Changelog Support/Training FAQ Toolbox

The Functional toolkit for Kotlin HTTP applications

http4k provides a simple and uniform way to serve, consume, and test HTTP services.

Quick Start

HTTP as a Function

```
val app = { request: Request ->
    Response(OK)
        .body("Hello, ${request.query("name")}!")
}

app.asServer(Jetty(9000)).start()
```

```
val client = ApacheClient()

val request = Request(GET, "http://localhost:9000")
                .query("name", "John Doe")

val response: Response = client(request)
```

Server as a Function. In Kotlin. Typesafe. Without the Server.

november 2017 / @daviddenton

Meet http4k

http4k is an HTTP toolkit written in Kotlin that enables the serving and consuming of HTTP services in a functional and consistent way.

Whenever (yet another) new JVM HTTP framework is released, the inevitable question that rightly get asked is **"How it this different to X?"**. In this post, I'm going to briefly cover what http4k is, how we think it's different, and address some of those **bold claims** from the title of this post.

Here's a quick rundown of what we think those differences are:

- http4k is small. Written in pure, functional Kotlin, with zero dependencies.
- http4k is simple. Like, really simple. No static API magic, no annotations, no reflection.
- http4k is immutable. It relies on an immutable HTTP model, which makes it a snap to test and debug.
- http4k is symmetric. It supports remote calls as a first-class concern, and the remote HTTP model is identical to the incoming HTTP model.
- http4k is typesafe. Say goodbye to all your validation and marshalling boilerplate and hello to automatic request validation and data class-based contracts for HTTP bodies using the Lens API.
- http4k is serverless. Or rather - server independent. Test an app out of container and then deploy it into any supported local container with 1 LOC - or as a function into AWS Lambda.

Oh god, not another framework! Why does this even exist?!?

Firstly - we don't consider http4k to be a framework - it's a set of libraries providing a functional

More unreads

#http4k

213 | 2 | http://www.http4k.org

Search

Thursday, February 13th

Jamie Hodkinson 09:49

Hi! We are responding with an input stream from a temporary file on disk.

```
Response(OK).body(tempFile.inputStream())
```

Is there any way to detect when the response has been fully read so that we can then delete the temp file on disk?

7 replies Last reply 12 days ago

Friday, February 14th

Mehdi Mohammadi 23:47

Could anybody help that what is wrong with this code?

```
 val app = { request: Request -> Response(OK).body("Hello, ${request.query("name")}!") }

val jettyServer = app.asServer(Jetty(9006)).start()
 val client = WebsocketClient.blocking(Uri.of("wss://echo.websocket.org"), headers = listOf(Pair("a
header", "value")))


        client.send(WsMessage("hello"))

        // read all of the messages from the socket until it is closed (by the server).
        // we expect to get one message back before the stream is closed.
        client.received().toList().forEach(::println)
```

I am able to connect but I am not receiving any message. I want to develop a websocket client always send and receive messages to/from the server (edited)

14 replies Last reply 1 day ago

Message #http4k

Aa

The End

 @dmitrykandalov

 github.com/dkandalov

 youtube.com/dkandalov

What about coroutine support? 

~~KT-18707~~ Created by Roman Elizarov 4 years ago Updated by Alexander Chernikov a month ago Visible to issue readers

Support suspend function as super type

43

The following shall be supported just like for regular functional types:

```
class MyFun : suspend () -> Result {
    suspend fun invoke(): Result { ... }
}
```

It is primarily needed for feature parity between regular and suspending functional types.

The use-case has recently came to light when adapting servlet streams to coroutine. There is a place where we need to change coroutine context to empty and restart a coroutine, and ability to implement a suspending functional type would have allowed to save one object (by implementing the corresponding functional type on a object that we are already creating).

Preview is available in Kotlin 1.5.30

Learn how to enable and use this feature in 1.5.30. Give it a try and share your feedback in this ticket!

Relates to 3 Is required for 1 Next step 1 Leads to 1

RELATES TO 3 ISSUES (1 UNRESOLVED)

- KT-46040 DM Apr 15 2021: Suspend function as super type
- ~~KT-46204~~ KJS / IR: Support `SuspendFunctionN` as super type 1
- ~~KT-15391~~ Prohibit suspend function type in supertype list 8

IS REQUIRED FOR 1 ISSUE (0 UNRESOLVED)

Project	Kotlin
Priority	Not specified
Type	Feature
Target versions	1.5.20, 1.6.0-RC2
State	Fixed
Assignee	Ilmir Usmanov
Subsystems	Language Design
Affected versions	No Affected versions
Fix in builds	1.5.20-dev-5783, 1.5.20-M1-27

Watchers 0 Stop watching

Boards Add to board

Closed **Coroutine support** #94

bdueck opened this issue on 28 Jan 2018 · 25 comments

daviddenton commented on 6 Feb • edited ▾ Collaborator

There is nothing going on with this at the moment. The core team have been working on other things (we all have actual paying jobs 😄), and this just hasn't gotten any attention because it's simply not the most important, or TBH, most interesting thing to be adding value to the library.

There are a few obstacles and barriers to our enthusiasm on this:

1. The core abstraction of HttpHandler would have to become an interface with a single suspend method instead of a typealias because fundamentally coroutines are a compiler trick and actually not supported by the typesystem - you cannot extend from a suspend function. We have modelled on a spike branch and it is quite easy to do, but it a large, breaking API change.
2. Once the core abstraction has been put in, we need to work out how to cleanly and simply integrate the coroutine interface with the various servers (and clients) - and there are 17 servers in total to do including the servlet and 6 serverless platforms. Current number of offers of assistance - **0**.
3. Loom - we are unsure of how this might affect the advantage for coroutines on the JVM at the top level.
4. Multiplatform - this would actually be a reason to move on with the work, but we are still tied to the JVM by the lack of stable IO and DateTime libraries.
5. Dependencies. Adding coroutines would actually massively increase the weight of the library and bring in several external libraries.
6. At the moment our focus is actually on Serverless platforms (and associated lightweight) and there are currently zero use-cases for coroutines there at the moment AFAIK.

So - just to bring clarity to this issue, I'm just going to close it for now until either lots of the above things are fixed, or someone actually starts paying us to deliver them.

> I'm also wondering, it does not seem entirely future-proof to adopt a framework which is based on blocking calls 2021.

That's entirely your call, but I'd encourage you not to be swayed by the web-scale hype. The initial use-case of Http4k was to run traffic for one of the top 1200 websites in the world, on which it serves a metric load of traffic quite easily on a few nodes (I think they were Jetty IIRC).