Komparing Kotlin Server Frameworks

Ken Yee @KAYAK
(Android and occasional backend developer)
KotlinConf 2018

Agenda

- What is a backend?
- What to look for in a server framework?
- What Kotlin frameworks are available?
- Pros/Cons of each framework
- Avoiding framework dependencies
- Serverless

What is a Backend?

REST API

Web server

Chat server

1. Backends are

What apps/clients talk to so that users can

- ➔ **Read dynamic data**
  So you can share information
- ➔ **Authenticate**
  Because it's about user access
- ➔ **Write persistent data**
  To save user interactions

2. Backends must

Be reliable

- ➔ **Read dynamic data**
  Scalable from user load
- ➔ **Authenticate**
  Secure from hacking
- ➔ **Write persistent data**
  Resilient to server failures

What do you look for in a framework?

Kotlin, DSL, Websockets, HTTP/2, Non-Blocking, CORS, CSRF, OIDC, OAuth2, Testing, Documentation

1. Kotlin!

On the server is:

- ➔ **Isomorphic Language**
  With Kotlin clients
- ➔ **Concise and Modern**
  Extension and Higher Order functions, DSLs, Coroutines
- ➔ **Null/Type Safe**
  Versus Javascript, Python, Ruby
- ➔ **Compatible w/ Java8**
  Delay moving to Java 9/10/11

Java (Spring)

```
public class BlogRouter {
        public RouterFunction<ServerResponse>
route(BlogHandler blogHandler) {
                return RouterFunctions
.route(RequestPredicates.GET("/blog").and(RequestPredicat
es.accept(MediaType.TEXT_HTML)),
blogHandler::findAllBlogs)

.route(RequestPredicates.GET("/blog/{slug}").and(RequestPr
edicates.accept(MediaType.TEXT_HTML)),
blogHandler::findOneBlog)

.route(RequestPredicates.GET("/api/blog").and(RequestPredi
cates.accept(MediaType.APPLICATION_JSON)),blogHandle
r::findOne)

.route(RequestPredicates.GET("/api/blog/{id}").and(Request
Predicates.accept(MediaType.APPLICATION_JSON)),
blogHandler::findOne);
        }
}
```

Kotlin (Spring)

```
class BlogRouter(private val blogHandler:
BlogHandler) {
        fun router() =
  router {
    ("/blog" and accept(TEXT_HTML)).nest {
        GET("/", blogHandler::findAllBlogs)
        GET("/{slug}",
blogHandler::findOneBlog)
    }

    ("/api/blog" and
accept(APPLICATION_JSON)).nest {
        GET("/", blogHandler::findAll)
        GET("/{id}", blogHandler::findOne)
    }
  }
}
```

Express.js

```
var router = express.Router()
var blogHandler = BlogHandler()

router.get('/blog', function (req, res) {
  res.send(blogHandler.findAllBlogs())
})
router.get('/blog/:slug', function (req, res) {
  res.send(blogHandler.findOneBlog(req))
})

router.get('/api/blog', function (req, res) {
  res.send(blogHandler.findAll())
})
router.get('/blog/:id', function (req, res) {
  res.send(blogHandler.findOne(req))
})
```

Kotlin (Spring)

```
class BlogRouter(private val blogHandler:
BlogHandler) {
        fun router() =
  router {
    ("/blog" and accept(TEXT_HTML)).nest {
        GET("/", blogHandler::findAllBlogs)
        GET("/{slug}",
blogHandler::findOneBlog)
    }

    ("/api/blog" and
accept(APPLICATION_JSON)).nest {
        GET("/", blogHandler::findAll)
        GET("/{id}", blogHandler::findOne)
    }
  }
}
```

2. Speed

Efficiency

- **Non-blocking**
  Reactor/RxJava/Kotlin Coroutines
  Event driven w/ Netty vs. threading
- **Http/2**
  Formerly Google's SPDY that uses single connections for multiple downloads
  Push resources to clients
- **Websockets**
  Useful for real-time chat/games

3. CORS (web)

Cross Origin Resource Sharing

- **Browser security**
  Allows browsers to access different domains for static files (images, CSS, JS, HTML, etc.)
- **Javascript security**
  Allows web clients to use different hosts for APIs

4. CSRF (web)

Cross Site Request Forgery

- **Browser form security**
  Prevents other sites from sending in the same form fields for a request
- **Javascript API call security**
  Prevent AJAX calls from being run from malicious sites

5. OAuth2/OIDC

Delegation vs. Authentication

- **Oauth2**
  Standard refresh tokens and access token that can be revoked
  Only for delegation
- **OIDC**
  OpenID Connect; aka, OAuth2 v2 or OpenID v3
  User verification
  JSON Web Token encoded data
  Auth token and delegation token
  Keycloak, Hydra, Okta, Auth0

6. Testing

Bug prevention

- **Unit Testing**
  Test internal business logic
- **Integration Testing**
  Test server integration

7. Documentation

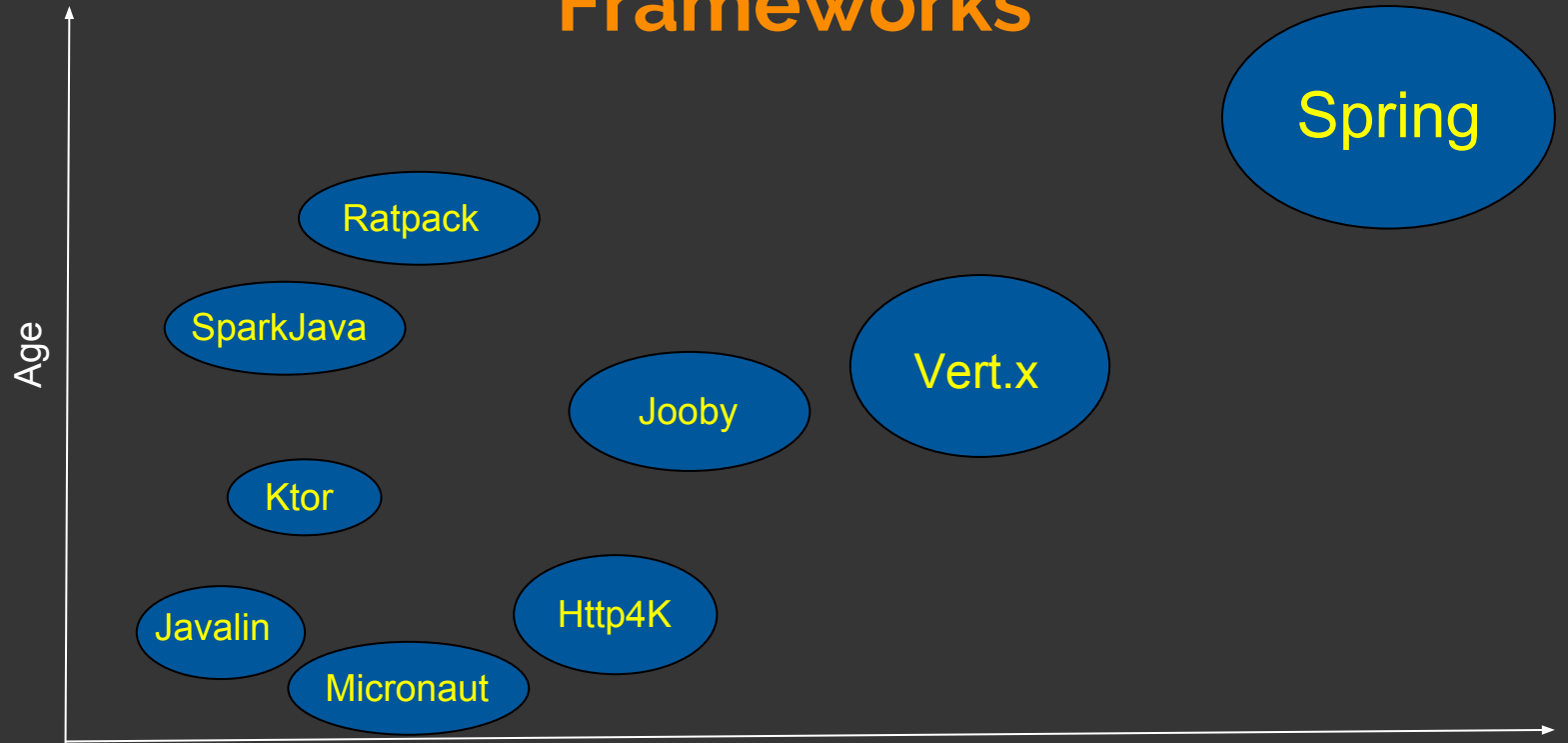
How, Help

- **Official Documentation**
  Clear documentation for features
  Useful examples
- **Community**
  StackOverflow
  Github stars
  Real projects
- **API**
  Swagger/RAML

Which frameworks?

Ktor, Http4K
Jooby
Vert.x
Spring, ...

Frameworks


Spring
Ratpack
SparkJava
Vert.x
Jooby
Ktor
Javalin
Http4K
Micronaut
Age
Features

Ktor

➔ **Pros**

JetBrains' official server framework
Pure Kotlin
Goal of Kotlin/Native support
Kotlin coroutine support
Websockets
Uses Netty for async engine
Includes Multi-platform client
Oauth support for Google/Twitter/FB

➔ **Cons**

Least mature pure-Kotlin framework
Not much module support but enough
Only Freemarker/Velocity templates
No opentracing/micrometer support

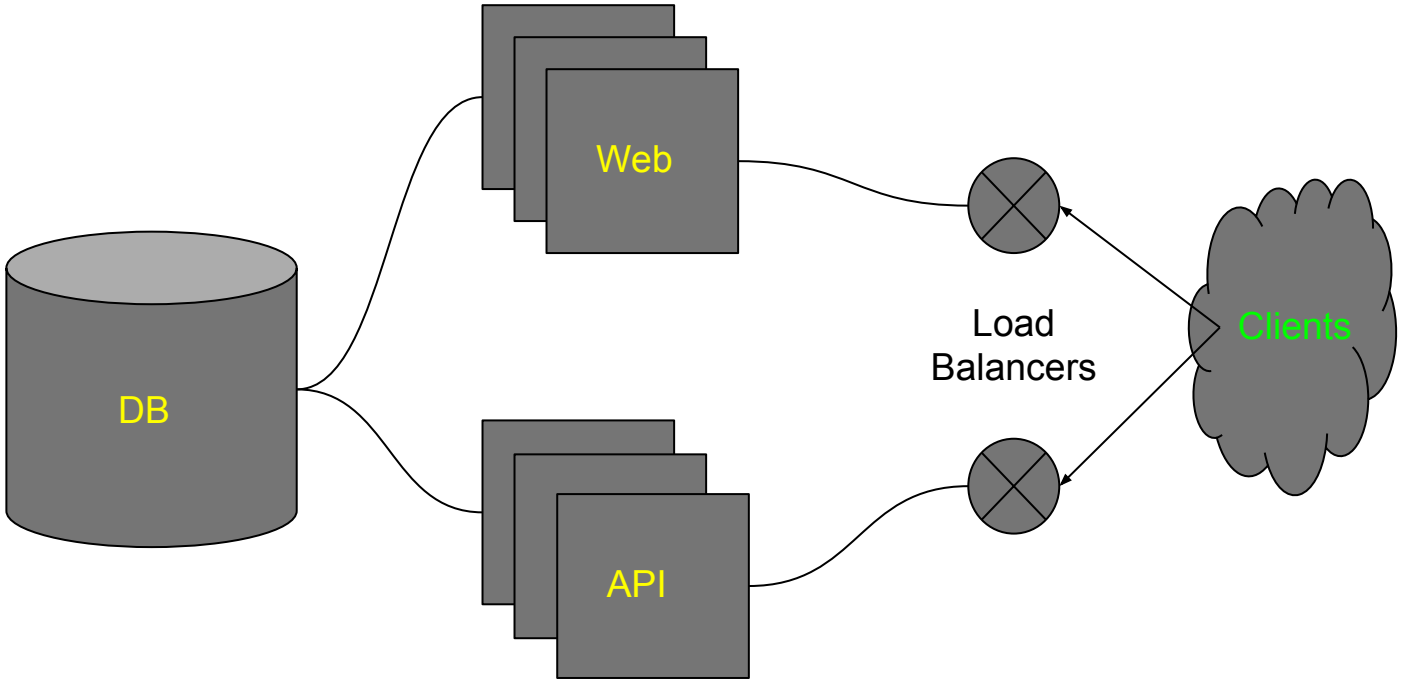
Ktor Routing

```
routing {
  accept(ContentType.Text.Html) {
    get("/blog") {
      call.respond(blogHandler::findAllBlogs)
    }
    get("/blog/{slug}") {
call.respond(blogHandler::findOneBlog)
    }
  }

  accept(ContentType.Application.Json) {
    get("/api/blog") {
      call.respond(blogHandler::findAll)
    }
    get("/api/blog/{id}") {
call.respond(blogHandler::findOne)
    }
  }
}
```

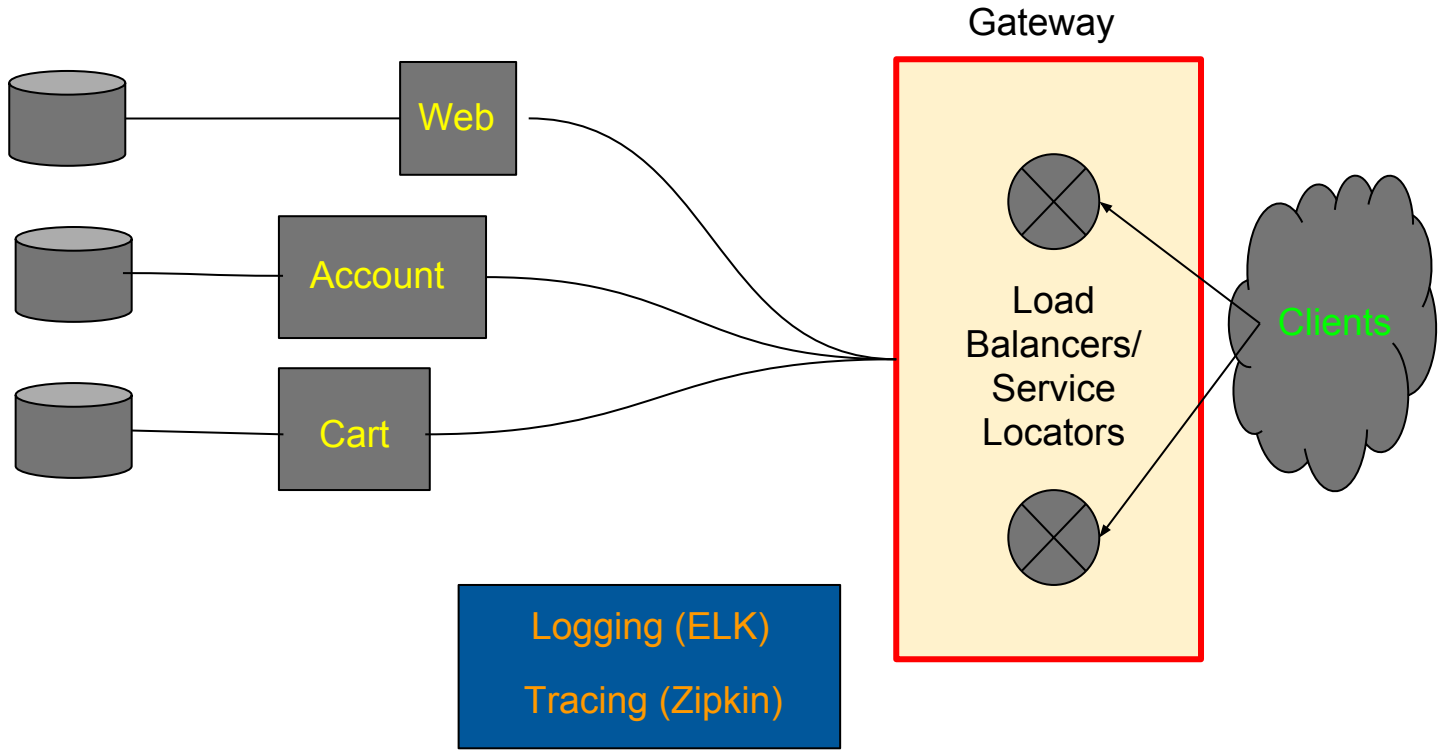
Ktor Coroutines

```
get("/{...}") {
  withContext(CommonPool) {
    call.slow()
  }
 }

private suspend fun ApplicationCall.slow() {
  respondHtml {
    body {
      "Slow result"
    }
  }
}
```

Monolith Architecture

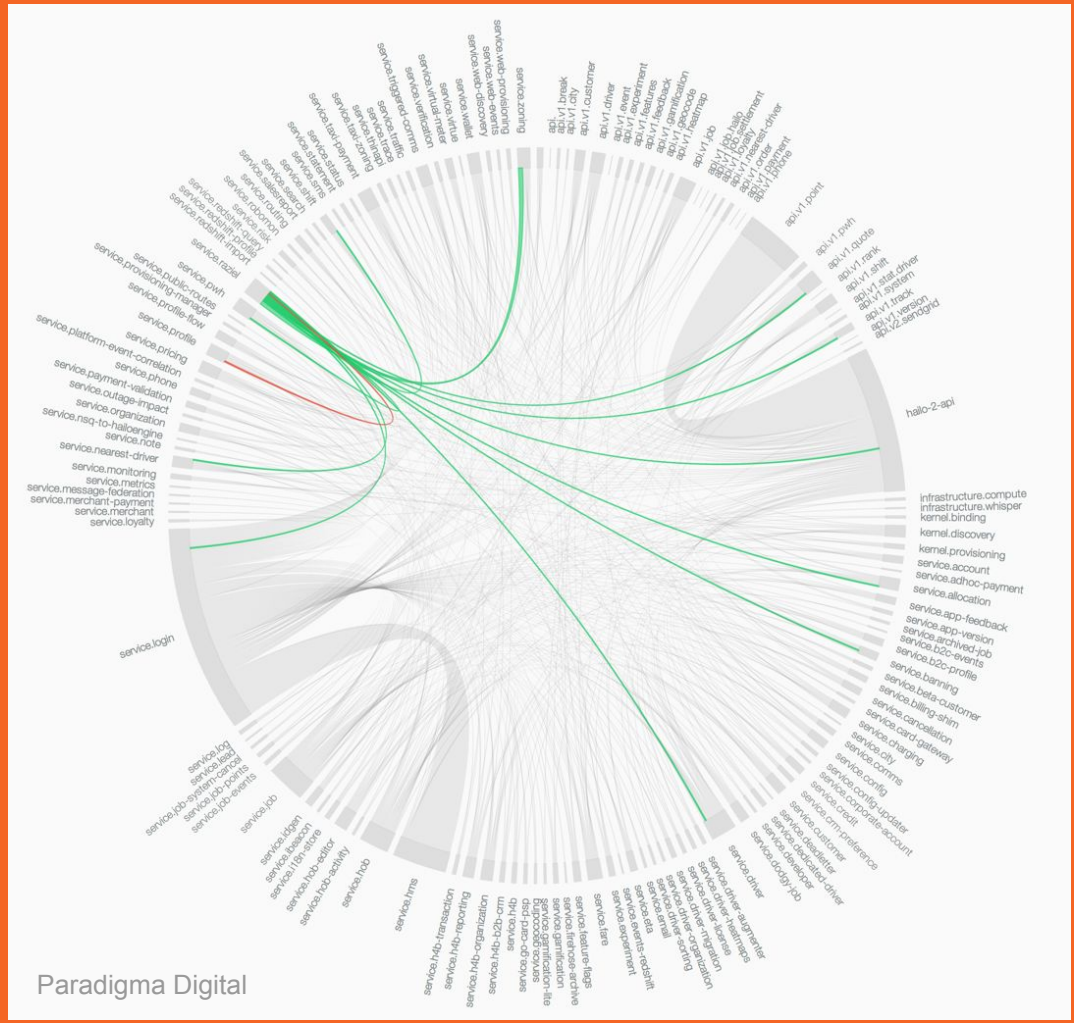


Microservice Architecture



Service Mesh

- ➔ **Massively Distributed**
- ➔ **Self-Healing**
- ➔ **Self-Scaling**
- ➔ Debugging is hard
- ➔ Logging is hard
- ➔ Perf monitoring is hard



Http4K

➔ **Pros**

Pure Kotlin
No magic reflection/annotations
Resilience4J support
Pluggable backends (Netty/Undertow)
Micrometer support
Swagger support
OAuth support for Auth0 and Google
Can deploy to AWS Lambda
GraalVM support
Chaos testing

➔ **Cons**

No Kotlin coroutine support
No Opentracing but has Zipkin
No auto JSON encode/decode

Http4K Routing

```
routes(
    “/blog” bind routes(
      “/” bind GET to { _ -> bloghandler.findAllBlogs()
},
      “/{slug}” bind GET to { req ->
bloghandler.findOneBlog(req) }
    ),
    “/api” bind routes(
      “/blog” bind GET to { _ -> bloghandler.findAll()
},
      “/blog/{id}” bind GET to { req ->
bloghandler.findOne(req) }
    )
).asServer(Jetty(8000)).start()
```

Jooby

➔ **Pros**

Pluggable backends
Event loop non-blocking
RxJava/Reactor
Even more modules than Http4K
Swagger/RAML
Lots of DB support
Job scheduling

➔ **Cons**

No Kotlin coroutine support
No zipkin or opentracing support

Jooby Routing

```
class App: Kooby({
  use(Jackson())

  get("/blog") {
    bloghandler.findAllBlogs()
  }
  get("/blog/:slug") { req ->

bloghandler.findOneBlog(req.param(“slug”).value)
  }

  get("/api/blog") {
    bloghandler.findAll()
  }
  get(“/api/blog/:id”) {
    blogHandler.findOne(req.param<Int>(“id”))
  }
})
```

Vert.x

➔ **Pros**

Kotlin coroutine support
Event loop non-blocking
Near top in TechEmpower benchmarks
Micrometer and Hawkular
Auto-clustering
Polyglot (JS, Python, Clojure, Java, etc.)
Redpipe for RxJava
Kovert (convention vs. configuration)
Swagger support
GraalVM support coming soon

➔ **Cons**

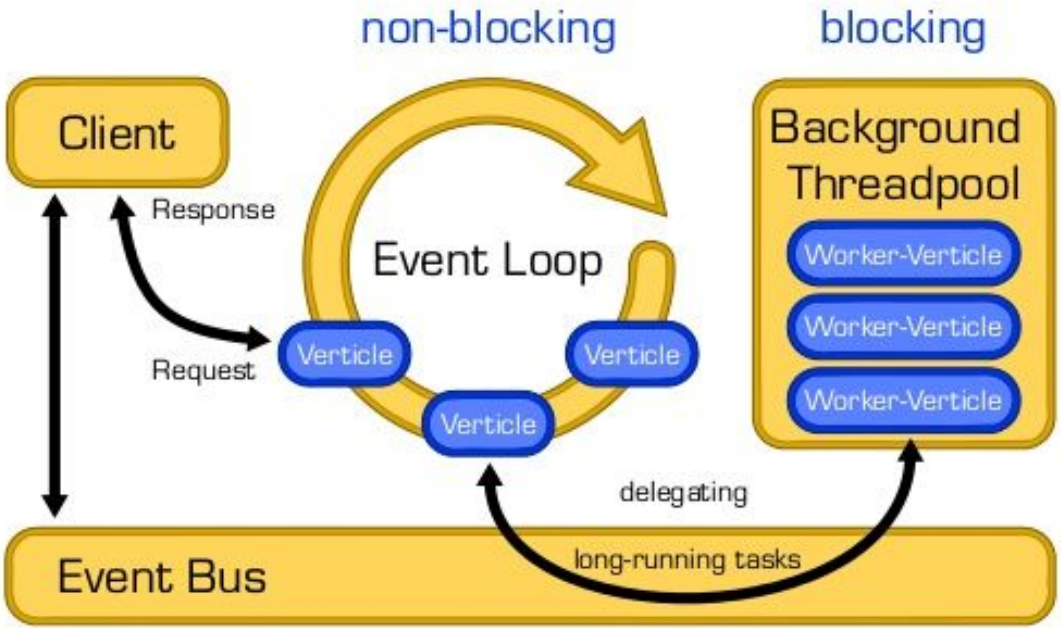
A bit more monolith than microservice
Not very mainstream in US
No auto JSON encode/decode

Vert.x Routing

```
private val router =
Router.router(vertx).apply {
  get("/blog")
    .handler(bloghandler::findAllBlogs)
  get("/blog/:slug")
    .handler(bloghandler::findOneBlog)

  get("/api/blog")
     .handler(bloghandler::findAll)
  get("/api/blog/:id")
    .handler (bloghandler::findOne)
}
```

Vert.x

Architecture



Vert.x Kovert

```
route.bindController(BlogApiController, “/api”)

class BlogApiController(val blogHandler: BlogHandler = Injekt.get())
{
  public fun listBlog(): List<Blog> = blogHandler.findAll()
  public fun findBlogById(val id: String): Blog = {
    val blog = blogHandler.findOne(id)
    if (blog == null) throw HttpErrorNotFound()
    return blog
}
```

Autogens:
/api/blog - list of blogs
/api/blog/:id - get blog with id

Spring

➔ **Pros**

Most popular framework
Webflux/Reactor non-blocking
Kitchen sink can be daunting
Spring Initializer project autogen
JHipster
Swagger support
GraalVM support in 5.2

➔ **Cons**

Need kotlin-spring/jpa plugins
No official Kotlin coroutine support (see Kaminski's coroutine library - Spring Fu)
Slowest framework

Spring Routing

```
class BlogRouter(private val blogHandler:
BlogHandler) {
        fun router() =
  router {
    ("/blog" and accept(TEXT_HTML)).nest {
        GET("/", blogHandler::findAllBlogs)
        GET("/{slug}",
blogHandler::findOneBlog)
    }

    ("/api/blog" and
accept(APPLICATION_JSON)).nest {
        GET("/", blogHandler::findAll)
        GET("/{id}", blogHandler::findOne)
    }
  }
}
```

Spring WebFlux

```
    @GetMapping("/api/blog/{id}")
    public Mono<ResponseEntity<Blog>>
getBlogById(@PathVariable(value = "id") String blogId) {
        return Mono.fromCallable(blogHandler.findOne(blogId))
                .map(blog -> ResponseEntity.ok(blog))
                .defaultIfEmpty(ResponseEntity.notFound().build());
    }
```

Spring Fu (incubator)

```
fun routes(blogHandler: BlogHandler) = router {
   "/blog".nest {
     GET("/") { blogHandler::findAllBlogs }
     GET("/{id}") { blogHandler::findOneBlog }
    }
    "/api/blog".nest {
        GET("/", blogHandler::findAll)
        GET("/{id}", blogHandler::findOne)
   }
}
```

JHipster

➔ **Pros**

Scaffolds Spring/Angular projects
Netflix OSS + Spring Boot
Can generate Kotlin projects
Design data models and autogen CRUD
RAILS/GRAILS-like
Generates Netflix microservice arch
Includes user management (UAA)

➔ **Cons**

Harder to find where to change things
Easy to complicate simple projects

JHipster Cmds

```
jhipster --blueprint generator-jhipster-kotlin
yo jhipster:import-jdl blog-jdl.jh
```

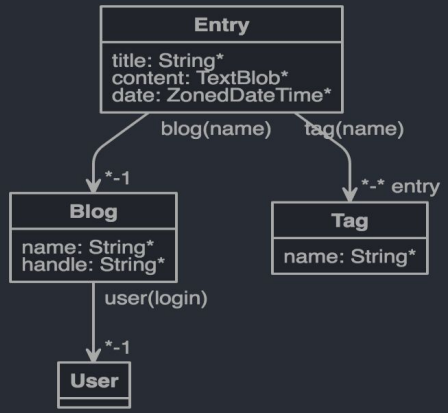
https://developer.okta.com/blog/2018/03/01/develop-microservices-jhipster-oauth

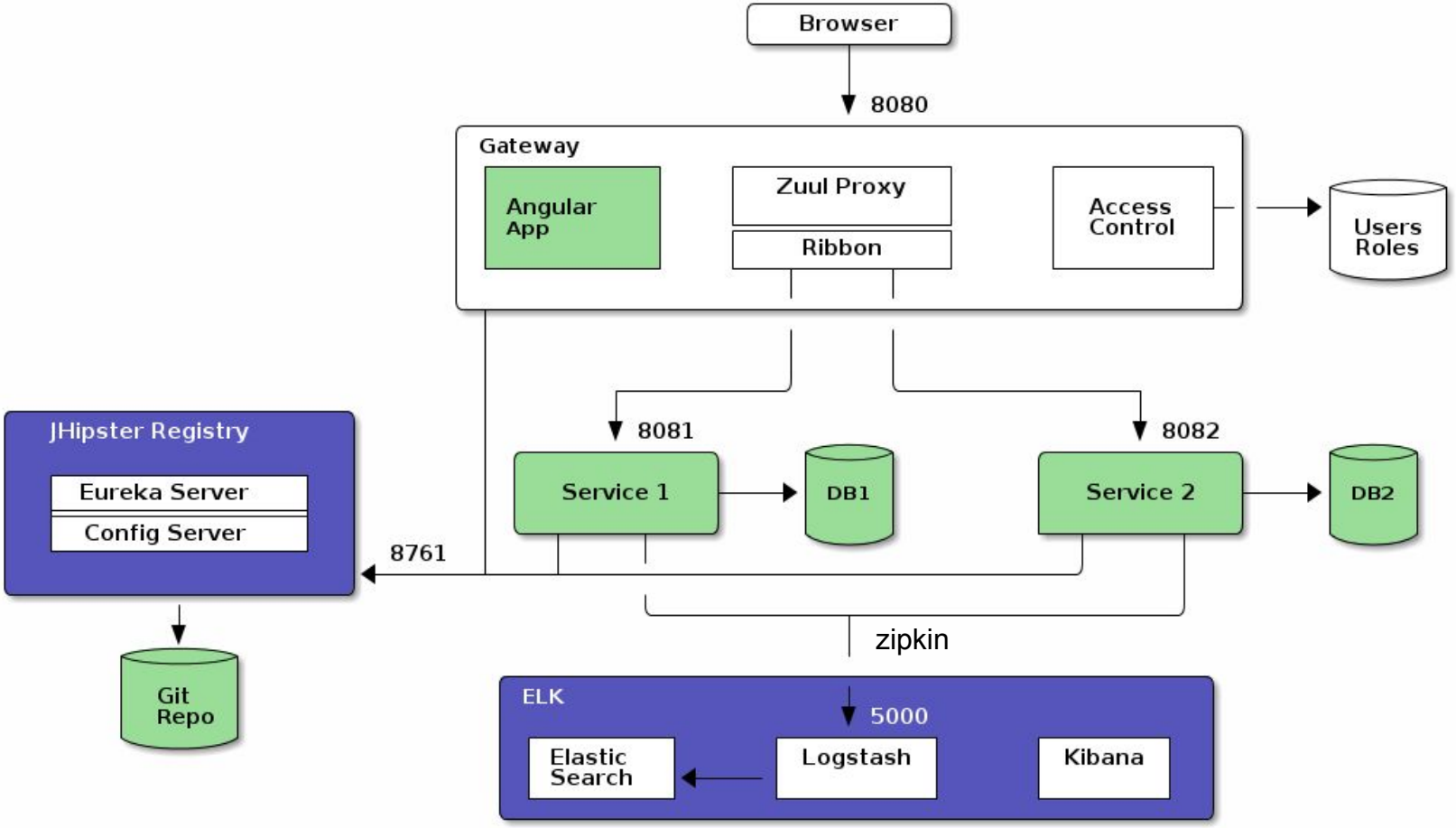
Secure https://start.jhipster.tech/jdl-studio/

JDL-Studio

Please sign in for mo

```
entity Blog {
    name String required minlength(3),
    handle String required minlength(2)
}

entity Entry {
    title String required,
    content TextBlob required,
    date ZonedDateTime required
}

entity Tag {
    name String required minlength(2)
}

relationship ManyToOne {
    Blog{user(login)} to User,
    Entry{blog(name)} to Blog
}

relationship ManyToMany {
    Entry{tag(name)} to Tag{entry}
}

paginate Entry, Tag with infinite-scroll
```




Browser
8080
Gateway
Angular App
Zuul Proxy
Ribbon
Access Control
Users Roles
JHipster Registry
Eureka Server
Config Server
8761
8081
Service 1
DB1
8082
Service 2
DB2
zipkin
Git Repo
ELK
5000
Elastic Search
Logstash
Kibana

Similarity?

DSL

Request headers/params

Response output

Context

Code Portability

- **Isolate Framework APIs**
  Business logic -> framework API
- **Isolate Business Logic**
  Business logic -> API facades
- **Isolate Database Access**
  Jooq or other DB layer
- **Coroutines or RxJava**
  kotlinx-coroutines-rx2

API Facades

- **Logging Facade**
  Slf4j
- **Tracing Facade**
  Opentracing, OpenCensus
- **Metrics Facade**
  Micrometer, OpenCensus, DW

Serverless?

Google Cloud Functions

AWS Lambda

OpenFAAS

Azure Functions

JVM Limitations

- ➔ Slow Startup
- ➔ Memory Usage
- ➔ Disk Size

How bad is it?

Compared to node.js

- ➔ **2-3x slower startup**
- ➔ **1.5-2x memory**
- ➔ **5-10x disk space**
  Including JS libs
- ➔ **2x cost on Amazon Lambda**

https://www.bouvet.no/bouvet-deler/comparing-java-and-node.js-on-aws-lambda

Kotlin vs. Go

- ➔ 2-3x slower startup
- ➔ 3x memory
- ➔ 2x disk space
- ➔ 8x cost on Amazon Lambda

https://blog.travelex.io/blazing-fast-microservice-with-go-and-lambda-d30d95290f28

But There's Hope

- ➔ **GraalVM (SubstrateVM)**
  Http4k, Vert.x, Spring 5.2
- ➔ **Kotlin/Native**
  Ktor

Limitations

GraalVM

➔ **No Dynamic Class Loading**

➔ **No JMX VM monitoring**

➔ Vert.x 3x less memory
Http4k 4x less memory

➔ Vert.x 9x faster startup
Http4k 50x faster startup

https://vertx.io/blog/eclipse-vert-x-goes-native/
https://www.richyhbm.co.uk/posts/compiling-kotlin-netty-webapp-with-graalvm/

All The Things!

Backends are easy in Kotlin

Lots of choices

Choose the features you need

Keep your business code portable

Further Reading

- https://nordicapis.com/api-security-oauth-openid-connect-depth/
- https://ktor.io/
- https://www.http4k.org/
- https://jooby.org/
- https://micronaut.io/
- https://vertx.io/
  - https://github.com/kohesive/kovert
  - http://redpipe.net/
- https://spring.io/
  - https://github.com/konrad-kaminski/spring-kotlin-coroutine
- https://www.jhipster.tech/
  - https://github.com/jhipster/jhipster-kotlin
- https://www.techempower.com/benchmarks/

KotlinConf Server Sessions

- Kotlin and Spring Boot
- Server as a Function (Http4K)
- Full Stack Kotlin on Google Cloud
- Building Server Backends with Ktor
- Painless Microservices with Kotlin
- GraphQL powered by Kotlin