Moving away from null and exceptions: An alternative way of error handling

Mario Fernandez

ThoughtWorks®

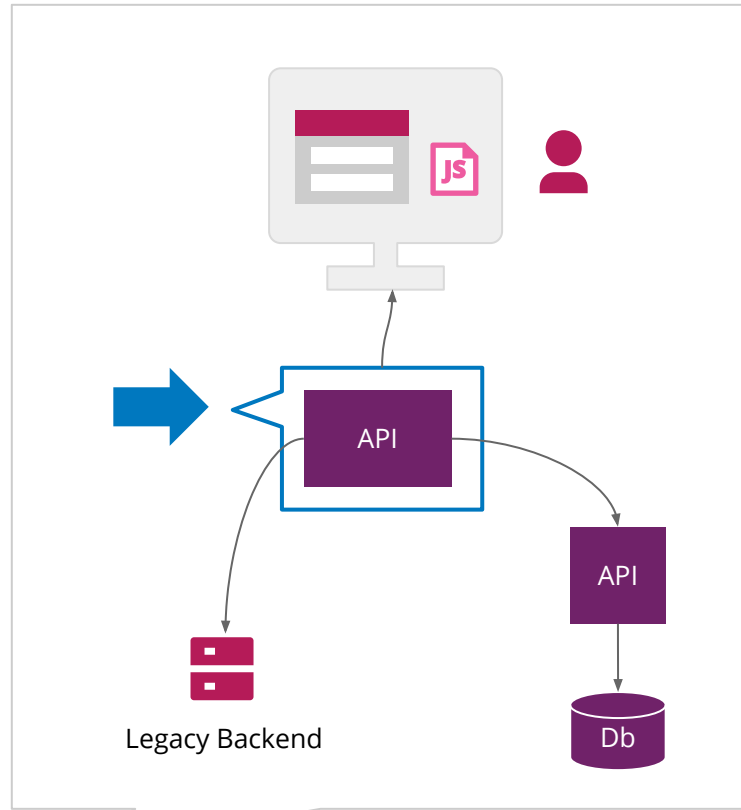# What to expect

The two takeaways

Let's stop **using null**

Let's stop **abusing** **exceptions**

Some Context

API

API

Legacy Backend

Db

Big German Corp

Null

```java
public static boolean isAdmin(List<Scope> scopes) {

    if(scopes == null) {

        return false;

    }


    Scope adminScope = findAdminScope(scopes);


    if(adminScope == null) {

        return false;

    }


    return adminScope.isValid();

}
```

Ad-hoc Error Handling

Null values sidestep the type system

Runtime instead of compile time feedback

Nullable Types
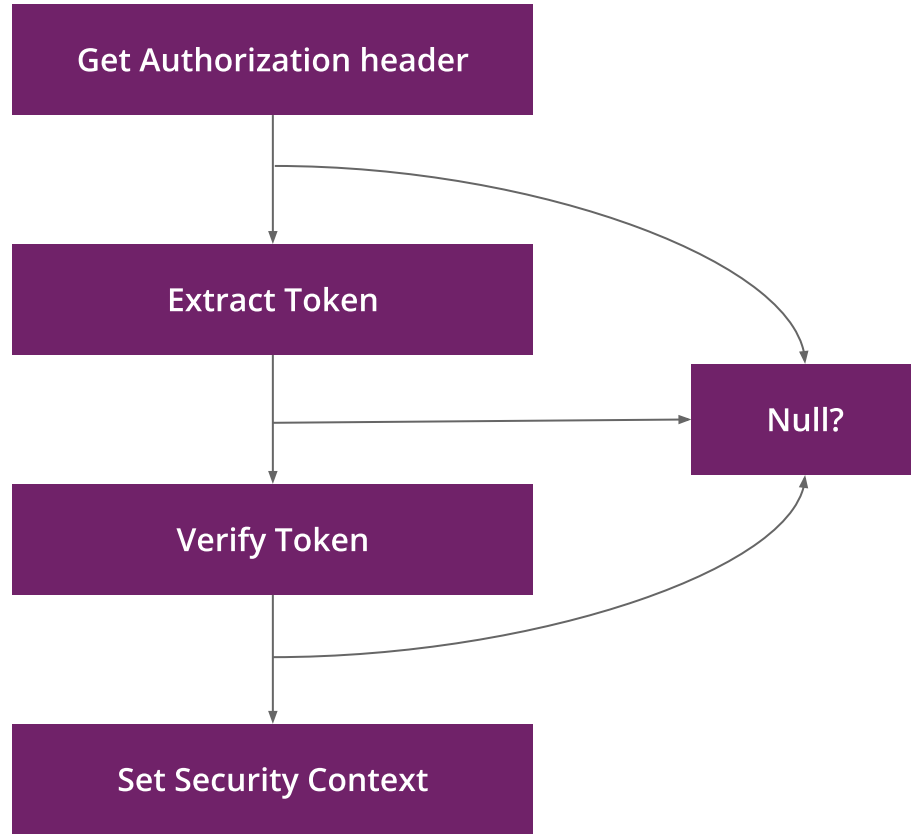
Authorization: Bearer bGciOi...JIUzI1NiIs

```kotlin
fun String.extractToken(): String? = if (startsWith("Bearer"))
    split(" ").last()
else
    null
```

```
header.extractToken()
  ?.let { token -> doStuff(token) }
```

What about more complex flows

```
request.getHeader(Headers.AUTHORIZATION)
  ?.let { header ->
    header.extractToken()
      ?.let { jwt ->
        verifier.verify(jwt)
          ?.let { token ->
            SecurityContextHolder.getContext().authentication = token
          }
      }
  }
```

Data Types

A digression about Functional Programming

A **data type** is an abstraction that encapsulates one reusable coding pattern

Think of **containers** holding your data

Provide clear **semantics** and an **interface**
to manipulate the data

Examples

Option   **Either**   Validated   IO

arrow-kt.io/docs/0.10/datatypes/intro/

Option

A value that might be absent

```kotlin
sealed class Option<out T> {
    data class Some<out T>(val a: T): Option<T>()
    object None: Option<Nothing>()
}
```

How to use it?

```kotlin
fun String.extractToken(): Option<String> = startsWith("Bearer ")
        .maybe { split(" ").last() }
```

```kotlin
when (val token = header.extractToken()) {
  is Option.None -> ResponseEntity.status(401).build()
  is Option.Some -> ResponseEntity.ok(result.t)
}
```

A container is not just a holder of data

```
interface Operations {
  fun <A, B> Option<A>.map(f: (A) -> B): Option<B>
  fun <A, B> Option<A>.flatMap(f: (A) -> Option<B>): Option<B>
}
```

Exceptions

The hidden GOTO

```
undertow.servlet.handlers.ServletChain$1.handleRequest(ServletChain.java:68)\n\tat io.undertow.servlet.handlers.ServletDispatchingHandler.handleRequest(Se
rvletDispatchingHandler.java:36)\n\tat io.undertow.servlet.handlers.RedirectDirHandler.handleRequest(RedirectDirHandler.java:68)\n\tat io.undertow.servle
t.handlers.security.SSLInformationAssociationHandler.handleRequest(SSLInformationAssociationHandler.java:132)\n\tat io.undertow.servlet.handlers.security.
ServletAuthenticationCallHandler.handleRequest(ServletAuthenticationCallHandler.java:57)\n\tat io.undertow.server.handlers.PredicateHandler.handleRequest
(PredicateHandler.java:43)\n\tat io.undertow.security.handlers.AbstractConfidentialityHandler.handleRequest(AbstractConfidentialityHandler.java:46)\n\tat
io.undertow.servlet.handlers.security.ServletConfidentialityConstraintHandler.handleRequest(ServletConfidentialityConstraintHandler.java:64)\n\tat io.unde
rtow.security.handlers.AuthenticationMechanismsHandler.handleRequest(AuthenticationMechanismsHandler.java:60)\n\tat io.undertow.servlet.handlers.security.
CachedAuthenticatedSessionHandler.handleRequest(CachedAuthenticatedSessionHandler.java:77)\n\tat io.undertow.security.handlers.AbstractSecurityContextAsso
ciationHandler.handleRequest(AbstractSecurityContextAssociationHandler.java:43)\n\tat io.undertow.server.handlers.PredicateHandler.handleRequest(Predicate
Handler.java:43)\n\tat io.undertow.server.handlers.PredicateHandler.handleRequest(PredicateHandler.java:43)\n\tat io.undertow.servlet.handlers.ServletInit
ialHandler.handleFirstRequest(ServletInitialHandler.java:269)\n\tat io.undertow.servlet.handlers.ServletInitialHandler.access$100(ServletInitialHandler.ja
va:78)\n\tat io.undertow.servlet.handlers.ServletInitialHandler$2.call(ServletInitialHandler.java:133)\n\tat io.undertow.servlet.handlers.ServletInitialHa
ndler$2.call(ServletInitialHandler.java:130)\n\tat io.undertow.servlet.core.ServletRequestContextThreadSetupAction$1.call(ServletRequestContextThreadSetup
Action.java:48)\n\tat io.undertow.servlet.core.ContextClassLoaderSetupAction$1.call(ContextClassLoaderSetupAction.java:43)\n\tat io.undertow.servlet.handl
ers.ServletInitialHandler.dispatchRequest(ServletInitialHandler.java:249)\n\tat io.undertow.servlet.handlers.ServletInitialHandler.access$000(ServletIniti
alHandler.java:78)\n\tat io.undertow.servlet.handlers.ServletInitialHandler$1.handleRequest(ServletInitialHandler.java:99)\n\tat io.undertow.server.Connec
tors.executeRootHandler(Connectors.java:376)\n\tat io.undertow.server.HttpServerExchange$1.run(HttpServerExchange.java:830)\n\tat java.base/java.util.conc
urrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1128)\n\tat java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.ja
va:628)\n\tat java.base/java.lang.Thread.run(Thread.java:834)\nCaused by: org.springframework.http.converter.HttpMessageNotReadableException: JSON parse e
rror: Unrecognized field \"businessPartnerId\" (c                                                      domain.VehicleConfiguration), not marked as igno
rable; nested exception is com.fasterxml.jackson.databind.exc.UnrecognizedPropertyException: Unrecognized field \"businessPartnerId\" (c
o                               s.domain.VehicleConfiguration), not marked as ignorable (0
l\", \"vehicleId\", \"orderType\", \"orderTypeLabel\", \"modelTypeLabel\"])\n at [Source: (PushbackInputStream); line: 1, column: 863] (through reference
ch                                                      .domain.VehicleConfiguration[\"businessPartnerId\"])\n\tat org.springframework.http.converter.js
on.AbstractJackson2HttpMessageConverter.readJavaType(AbstractJackson2HttpMessageConverter.java:245)\n\tat org.springframework.http.converter.json.Abstract
Jackson2HttpMessageConverter.read(AbstractJackson2HttpMessageConverter.java:227)\n\tat org.springframework.web.client.HttpMessageConverterExtractor.extrac
tData(HttpMessageConverterExtractor.java:104)\n\t... 105 common frames omitted\nCaused by: com.fasterxml.jackson.databind.exc.UnrecognizedPropertyExceptio
n: Unrecognized field \"businessPartnerId\" (c                                                      s.domain.VehicleConfiguration), not marked as ignorab
le (                                                                                                                    n at [Source:
(PushbackInputStream); line: 1, column: 863] (t
usinessPartnerId\"])\n\tat com.fasterxml.jackson.databind.exc.UnrecognizedPropertyException.from(UnrecognizedPropertyException.java:61)\n\tat com.fasterxm
l.jackson.databind.DeserializationContext.handleUnknownProperty(DeserializationContext.java:840)\n\tat com.fasterxml.jackson.databind.deser.std.StdDeseria
lizer.handleUnknownProperty(StdDeserializer.java:1206)\n\tat com.fasterxml.jackson.databind.deser.BeanDeserializerBase.handleUnknownProperty(BeanDeseriali
zerBase.java:1592)\n\tat com.fasterxml.jackson.databind.deser.BeanDeserializerBase.handleUnknownProperties(BeanDeserializerBase.java:1542)\n\tat com.faste
rxml.jackson.databind.deser.BeanDeserializer._deserializeUsingPropertyBased(BeanDeserializer.java:438)\n\tat com.fasterxml.jackson.databind.deser.BeanDese
rializerBase.deserializeFromObjectUsingNonDefault(BeanDeserializerBase.java:1287)\n\tat com.fasterxml.jackson.databind.deser.BeanDeserializer.deserializeF
romObject(BeanDeserializer.java:326)\n\tat com.fasterxml.jackson.databind.deser.BeanDeserializer.deserialize(BeanDeserializer.java:159)\n\tat com.fasterxm
l.jackson.databind.ObjectMapper._readMapAndClose(ObjectMapper.java:4202)\n\tat com.fasterxml.jackson.databind.ObjectMapper.readValue(ObjectMapper.java:325
```
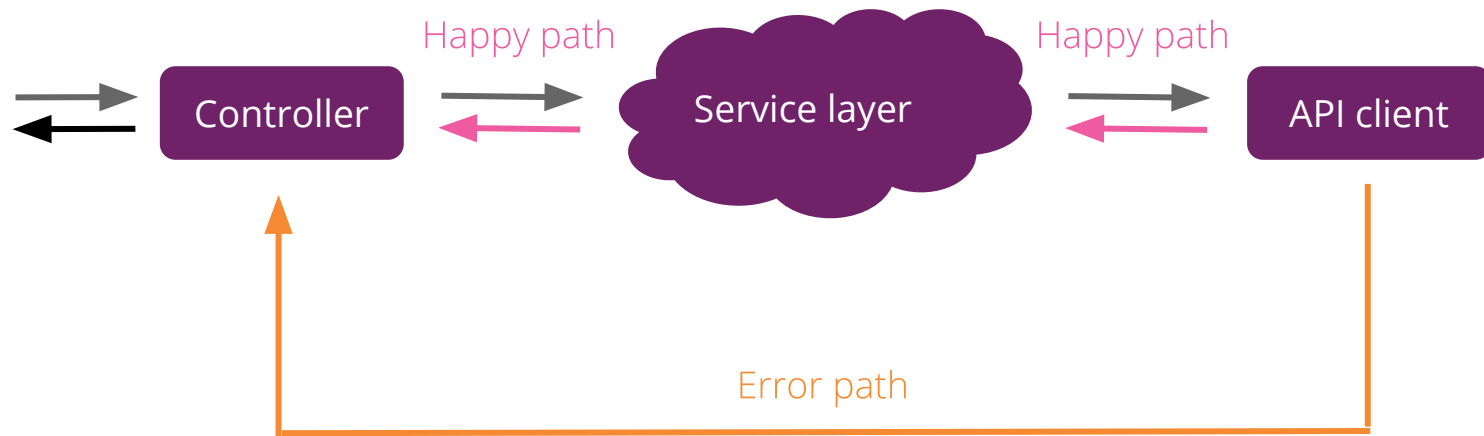
At first Exceptions may sound like
a good idea

Separate flow

Controller   Happy path   Service layer   Happy path   API client

Error path

Breaks encapsulation

```kotlin
interface Verifier {

    /**
     * @param jwt a jwt token
     * @return authentication credentials
     */
    fun verify(jwt: String): TokenAuthentication

}
```

```java
/**
 * Perform the verification against the given Token
 *
 * @param token to verify.
 * @return a verified and decoded JWT.
 * @throws AlgorithmMismatchException
 * @throws SignatureVerificationException
 * @throws TokenExpiredException
 * @throws InvalidClaimException
 */
public DecodedJWT verifyByCallingExternalApi(String token);
```

A typical way of handling this

```kotlin
@ExceptionHandler(JWTVerificationException::class)
fun handleException(exception: JWTVerificationException):
  ResponseEntity<ErrorMessage> {
    return ResponseEntity
      .status(HttpStatus.BAD_GATEWAY)
      .body(ErrorMessage.fromException(exception))
}
```

Either

Two different values depending on the result of the computation

Either<T>

Left
Exception

Right
T

```kotlin
sealed class Either<out L, out R> {
  data class Left<out L, out R>(val a: L) : Either<L, R>()
  data class Right<out L, out R>(val b: R) : Either<L, R>()
}
```

How to use it?

```kotlin
interface Verifier {
  fun verify(token: String): Either<TokenEx, TokenAuth>
}
```

```kotlin
fun Verifier.unsafeVerify(jwt: String): Either<TokenEx, TokenAuth> = try {

  verifyByCallingExternalApi(jwt).right()

} catch (e: JWTVerificationException) {

  e.left()

}
```

An evolving computation

```kotlin
interface Operations {
  fun <T, A, B> Either<T, A>.map(f: (A) -> B): Either<T, B>
  fun <T, A, B> Either<T, A>.flatMap(f: (A) -> Either<T, B>):
    Either<T, B>
}
```
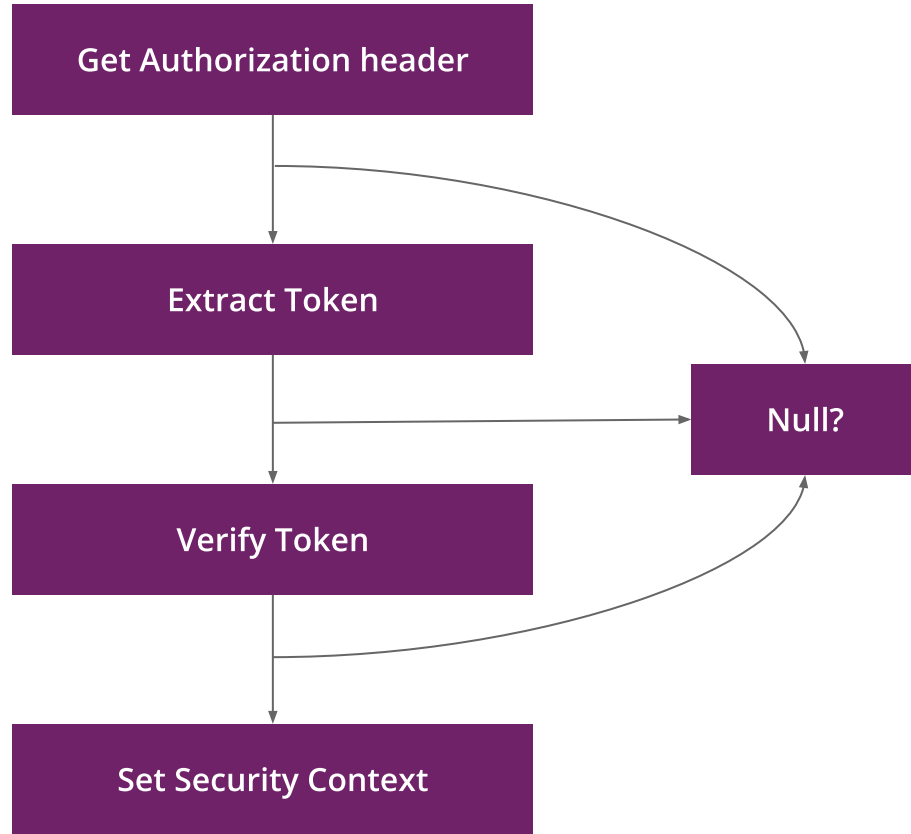
```
verifier
    .unsafeVerify(jwt)
    .map { it.asToken() }
```

What about more complex flows

```
request.getHeader(Headers.AUTHORIZATION)
  .toEither()
  .flatMap { header ->
    header.extractToken()
      .flatMap { jwt ->
        verifier
          .verify(jwt)
          .map { token ->
            SecurityContextHolder.getContext().authentication = token
          }
      }
  }
```

Non-Nested Syntax

Similar to **async/await**

```
Either.fx {
    val (header) = request.getHeader(Headers.AUTHORIZATION).toEither()
    val (jwt) = header.extractToken()
    val (token) = verifier.verify(jwt)
    SecurityContextHolder.getContext().authentication = token
}
```

thoughtworks.com/insights/blog/either-data-type-alternative-throwing-exceptions

Summary

Null and exceptions can lead to flaky, hard to understand code

Let's stop **using null**

Let's stop **abusing** **exceptions**