Kotlin
In Practice

@philipp_hauer
Spreadshirt

JUG Saxony Meetup, 25.10.18


Spreadshirt

Hands Up!

Kotlin Features and Usage in Practice

Data Classes

Immutability made easy

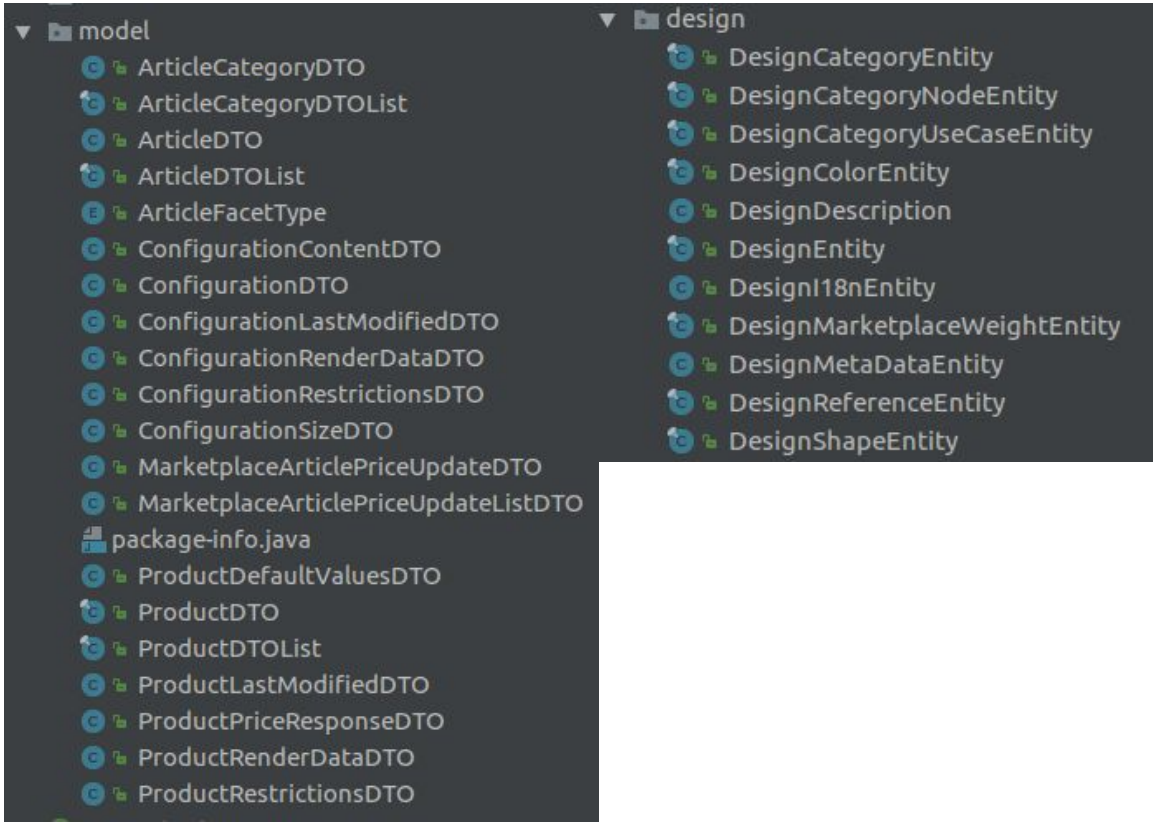
```
data class DesignData(
        val fileName: String,
        val uploaderId: Int,
        val width: Int = 0,
        val height: Int = 0
)
```

- Constructor (assign args to fields)
- Getter
- toString()
- hashCode(), equals()
- copy()
- Default Arguments (no chaining)

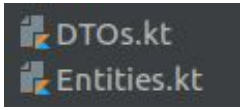
```
val design = DesignData(fileName = "cat.jpg", uploaderId = 2)

val fileName = design.fileName
design.fileName = "dog.jpg"

val design2 = design.copy(fileName = "dog.jpg")
```

Putting Classes Together

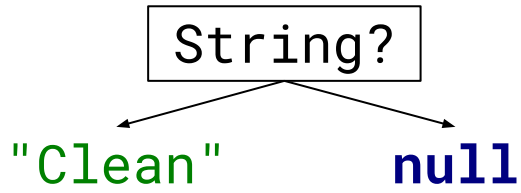
Java

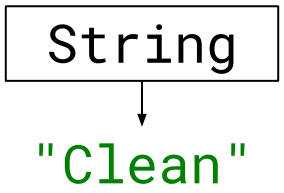


Kotlin



Null-Safety and Means for Null Handling






```
val value: String = "Clean Code"
val value: String = null

val nullableValue: String? = "Clean Code"
val nullableValue: String? = null
val v: String = nullableValue
```

smart-cast!

```
val v: String = if (nullableValue == null) "default" else nullableValue

val v: String = nullableValue ?: "default"
```

Null-Safety and Means for Null Handling

```
val city = order.customer.address.city
```

```
val city = order!!.customer!!.address!!.city
```

avoid this!

```
if (order == null || order.customer == null ||
   order.customer.address == null){
    throw IllegalArgumentException("Invalid Order")
}
val city = order.customer.address.city
```

smart-cast

```
val city = order?.customer?.address?.city
```

```
val city = order?.customer?.address?.city ?:
   throw IllegalArgumentException("Invalid Order")
```

No NullPointerExceptions in Production Anymore™

Expressions

Flow control structures are expressions!

```
val json = """{"message": "HELLO"}"""
val message = try {
    JSONObject(json).getString("message")
} catch (ex: JSONException) {
    json
}
```

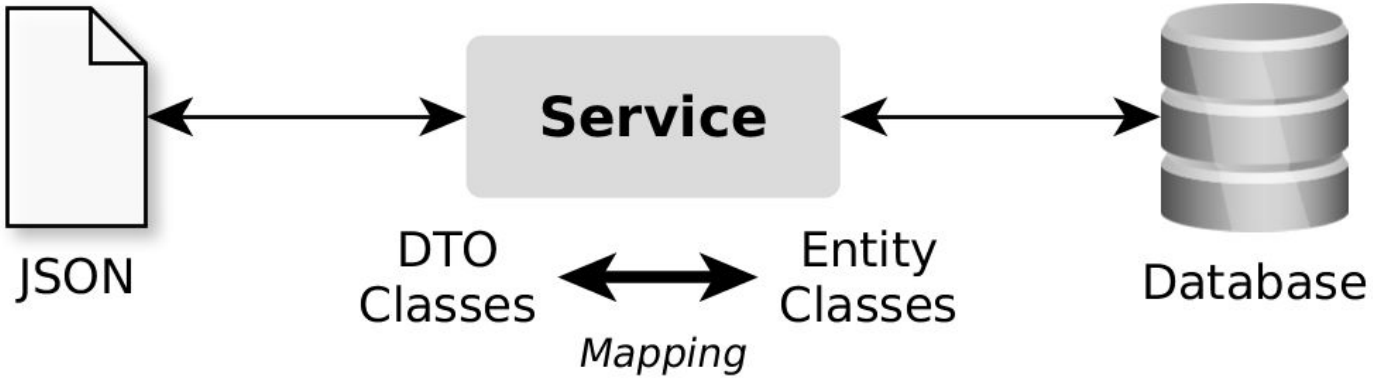
Expressions

Single Expression Functions

```
fun getMessage(json: String): String {
    val message = try {
        JSONObject(json).getString("message")
    } catch (ex: JSONException) {
        json
    }
    return message
}
```

```
fun getMessage(json: String) = try {
    JSONObject(json).getString("message")
} catch (ex: JSONException) {
    json
}
```

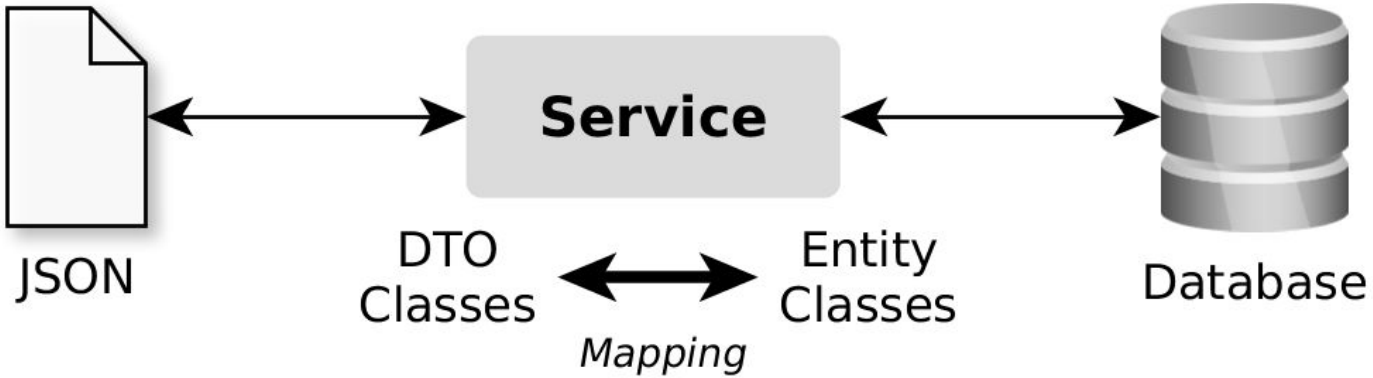
Concise Mapping between Model Classes




```
data class SnippetDTO(
    val code: String,
    val author: String,
    val date: Instant
)
```

```
data class SnippetEntity(
    val code: String,
    val author: AuthorEntity,
    val date: Instant
)
data class AuthorEntity(
    val firstName: String,
    val lastName: String
)
```

Concise Mapping between Model Classes




```
fun mapToDTO(entity: SnippetEntity) = SnippetDTO(
    code = entity.code,
    date = entity.date,
    author = "${entity.author.firstName} ${entity.author.lastName}"
)
```

Processing an HTTP Response in Java

```
public Product parseProduct(Response response){
    if (response == null){
        throw new ClientException("Response is null");
    }
    int code = response.code();
    if (code == 200 || code == 201){
        return mapToDTO(response.body());
    }
    if (code >= 400 && code <= 499){
        throw new ClientException("Sent an invalid request");
    }
    if (code >= 500 && code <= 599){
        throw new ClientException("Server error");
    }
    throw new ClientException("Error. Code " + code);
}
```

Processing an HTTP Response in Kotlin: when

```
fun parseProduct(response: Response?) = when (response?.code()){
    null -> throw ClientException("Response is null")
    200, 201 -> mapToDTO(response.body())
    in 400..499 -> throw ClientException("Sent an invalid request")
    in 500..599 -> throw ClientException("Server error")
    else -> throw ClientException("Error. Code ${response.code()}")
}
```

Do-It-Yourself ORM

```
class UserDAO(private val template: JdbcTemplate) {

   fun findAllUsers() = template.query("SELECT * FROM users;", this::mapToUser)

   fun findUser(id: Int): User? = try {
      template.queryForObject("SELECT * FROM users WHERE id = $id;", this::mapToUser)
   } catch (e: EmptyResultDataAccessException) {
      null
   }

   private fun mapToUser(rs: ResultSet, rowNum: Int) = User(
       email = rs.getString("email"),
       name = mergeNames(rs),
       role = if (rs.getBoolean("guest")) Role.GUEST else Role.USER,
       dateCreated = rs.getTimestamp("date_created").toInstant(),
       state = State.valueOf(rs.getString("state"))
   )
}
```

Spring: Easy Constructor Injection

```java
// Java
public class CustomerResource {

    private CustomerRepository repo;
    private CRMClient client;

    public CustomerResource(CustomerRepository repo, CRMClient client) {
        this.repo = repo;
        this.client = client;
    }
}
```

```kotlin
// Kotlin
class CustomerResource(private val repo: CustomerRepository,
                       private val client: CRMClient){
}
```

Concise Lambda Expressions & Vaadin

```
val button = Button("Delete")


button.addClickListener( { event -> println(event) } )

button.addClickListener { event -> println(event) }

button.addClickListener { println(it) }
```

Collection API

Read-only Collections

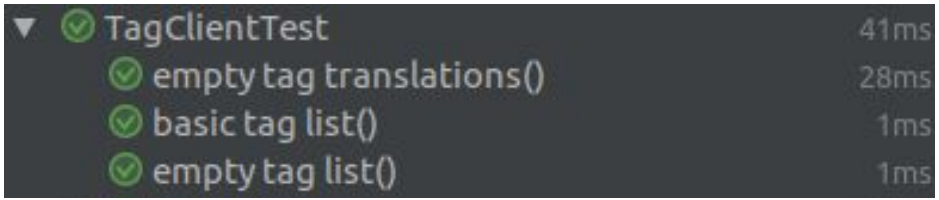
```
val list = listOf(1,2,3,4)
list.add(1)
```

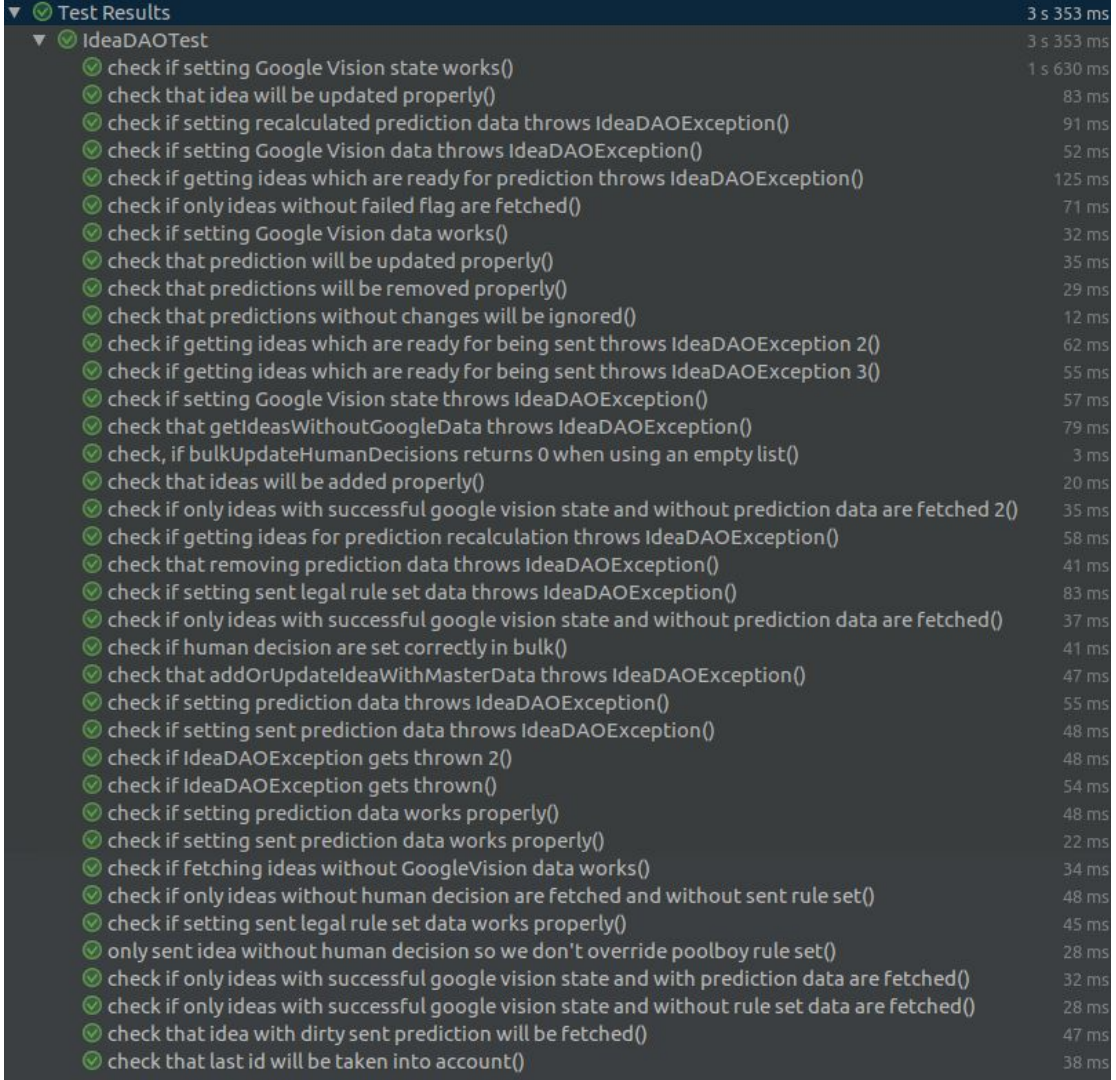
Collections API

```
val evenList = list.filter { it % 2 == 0 }

val daysList = list.filter { it % 2 == 0 }
                   .map { DayOfWeek.of(it) }
println(daysList) //[TUESDAY, THURSDAY]
```

Testing: Backticks

```
class TagClientTest {
    @Test
    fun `basic tag list`() {}
    @Test
    fun `empty tag list`() {}
}
```

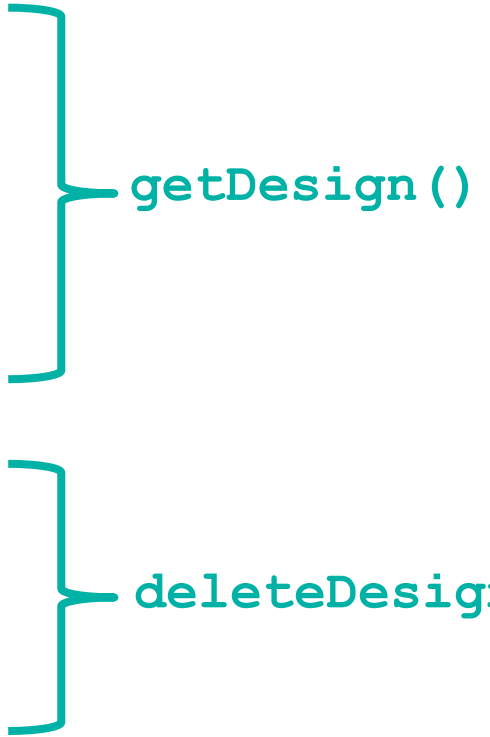


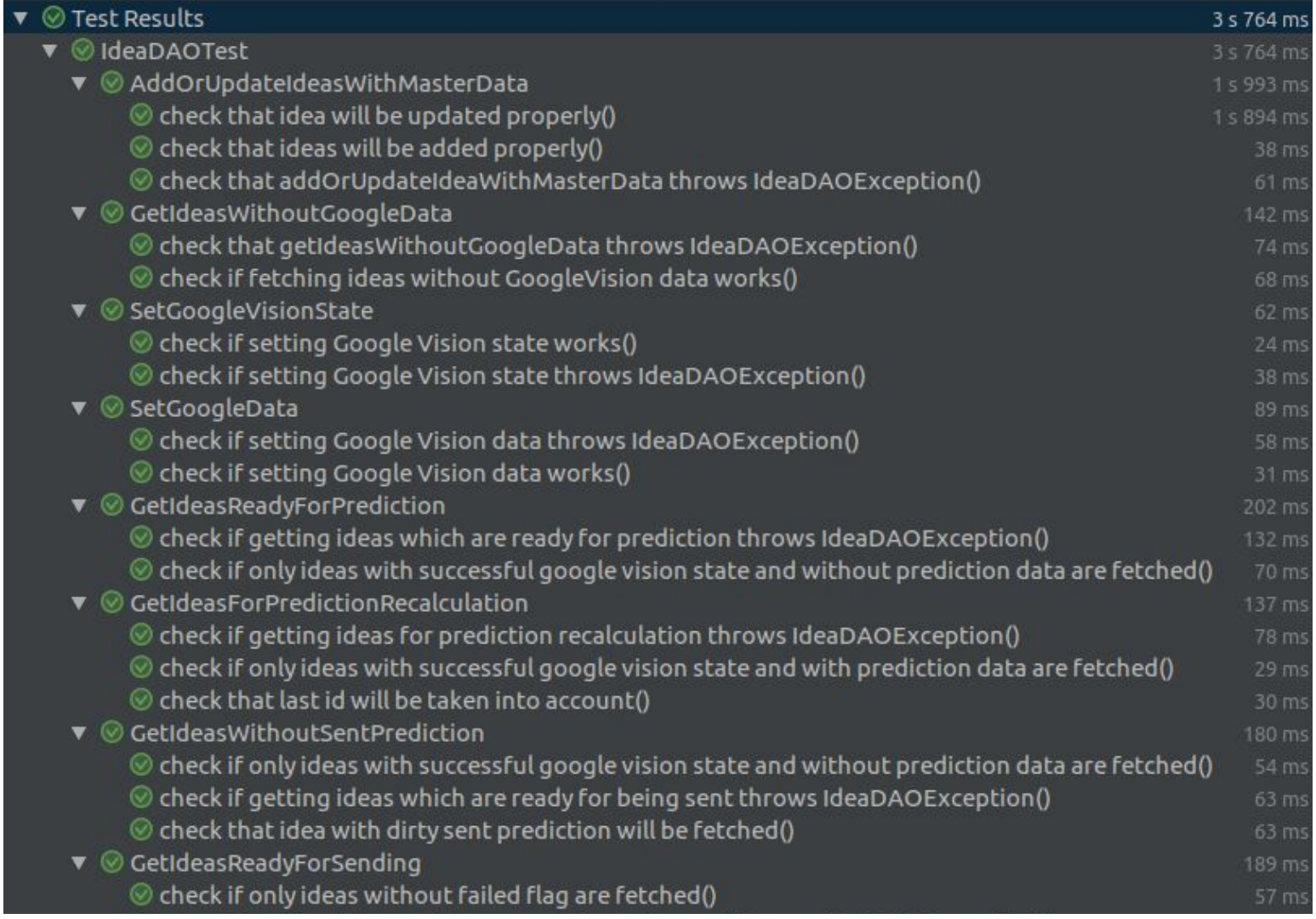


Which test belongs to which method?

Testing: @Nested Inner Classes

```
class DesignControllerTest {
    @Nested
    inner class GetDesigns {
        @Test
        fun `all fields are included`() {}
        @Test
        fun `limit parameter`() {}
    }
    @Nested
    inner class DeleteDesign {
        @Test
        fun `design is removed in db`() {}
    }
}
```




Test Results 3 s 764 ms
IdeaDAOTest 3 s 764 ms
AddOrUpdateIdeasWithMasterData 1 s 993 ms
check that idea will be updated properly() 1 s 894 ms
check that ideas will be added properly() 38 ms
check that addOrUpdateIdeaWithMasterData throws IdeaDAOException() 61 ms
GetIdeasWithoutGoogleData 142 ms
check that getIdeasWithoutGoogleData throws IdeaDAOException() 74 ms
check if fetching ideas without GoogleVision data works() 68 ms
SetGoogleVisionState 62 ms
check if setting Google Vision state works() 24 ms
check if setting Google Vision state throws IdeaDAOException() 38 ms
SetGoogleData 89 ms
check if setting Google Vision data throws IdeaDAOException() 58 ms
check if setting Google Vision data works() 31 ms
GetIdeasReadyForPrediction 202 ms
check if getting ideas which are ready for prediction throws IdeaDAOException() 132 ms
check if only ideas with successful google vision state and without prediction data are fetched() 70 ms
GetIdeasForPredictionRecalculation 137 ms
check if getting ideas for prediction recalculation throws IdeaDAOException() 78 ms
check if only ideas with successful google vision state and with prediction data are fetched() 29 ms
check that last id will be taken into account() 30 ms
GetIdeasWithoutSentPrediction 180 ms
check if only ideas with successful google vision state and without prediction data are fetched() 54 ms
check if getting ideas which are ready for being sent throws IdeaDAOException() 63 ms
check that idea with dirty sent prediction will be fetched() 63 ms
GetIdeasReadyForSending 189 ms
check if only ideas without failed flag are fetched() 57 ms

Top-Level Functions

```java
// StringUtils.java
public class StringUtils {
    public static String wrap(String value, String wrapWith) {
        return wrapWith + value + wrapWith;
    }
}
```

```kotlin
// StringUtils.kt
fun wrap(value: String, wrapWith: String): String {
    return wrapWith + value + wrapWith
}

// usage
import net.spreadshirt.*
wrap("Ahoi!", "*")
```

Extension Functions

```
// definition
fun String.wrap(wrapWith: String): String {
    return wrapWith + this + wrapWith
}

// usage
val wrapped = "hello".wrap("*")

// as opposed to:
val wrapped = StringUtils.wrap("hello", "*")
```

Extension Functions: Add UI Logic

```
enum class SnippetState{ EXECUTED, NOT_EXECUTED }

fun SnippetState.toIcon() = when (this){
    SnippetState.EXECUTED -> FontAwesome.THUMBS_O_UP
    SnippetState.NOT_EXECUTED -> FontAwesome.THUMBS_O_DOWN
}

//usage:
val icon = state.toIcon()
```

Extension Functions: Extend Libraries

```
assertThat(taxRate1).isCloseTo(0.3f, Offset.offset(0.001f))
assertThat(taxRate2).isCloseTo(0.2f, Offset.offset(0.001f))
assertThat(taxRate3).isCloseTo(0.5f, Offset.offset(0.001f))
```

Duplication

```
fun AbstractFloatAssert<*>.isCloseTo(expected: Float)
    = this.isCloseTo(expected, Offset.offset(0.001f))
```

```
// Usage:
assertThat(taxRate1).isCloseTo(0.3f)
assertThat(taxRate2).isCloseTo(0.2f)
assertThat(taxRate3).isCloseTo(0.5f)
```

Clean

Idiomatic

Kotlin at Spreadshirt

Ecosystem vs. Language

Evaluation of Kotlin

Pros

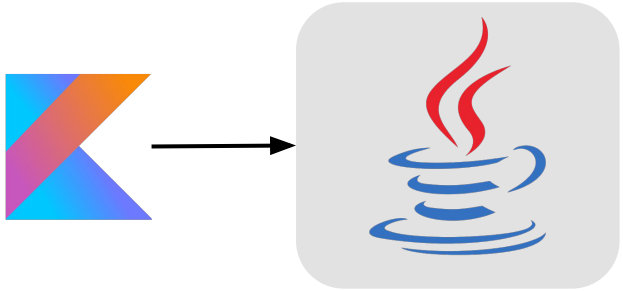
- **Reuse of the powerful and well-known Java ecosystem**
- Interoperability with Java.
- Productivity
- Less error-prone
- Easy to learn. No paradigm shift.
- Stepwise migration possible.
- Brilliant IDE support with IntelliJ IDEA.

Cons

- Training required
- **Further development depends on JetBrains.**
- **Poor Support for other IDEs (like Eclipse)**

⇒ Low Risks

Kotlin Usage at Spreadshirt



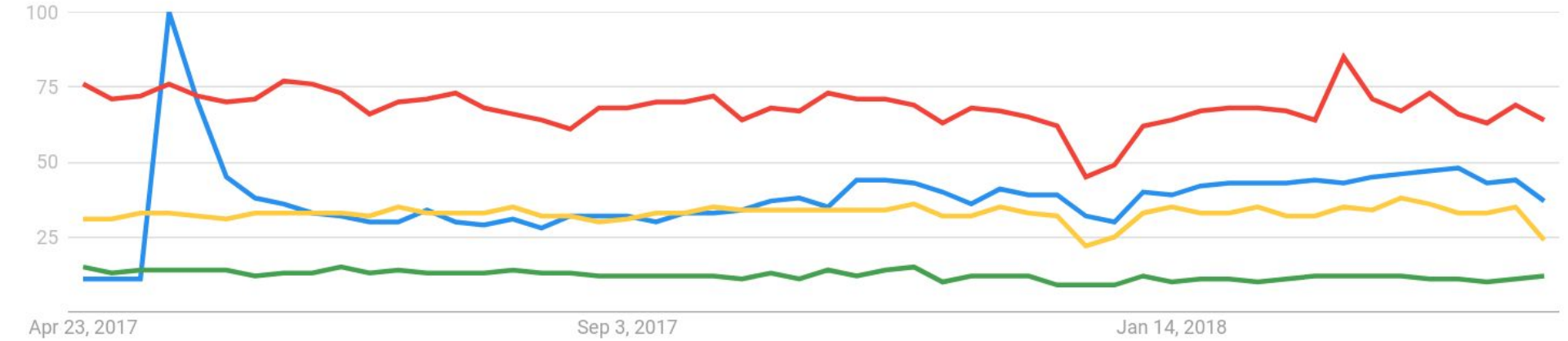
3 Test Projects

1 Java service enriched with Kotlin

15 new services and tools purely written in Kotlin

Adoption of Kotlin Today (Outside of Spreadshirt)

Google Search Trends



Scala Kotlin Groovy Clojure

Peak: Google I/O '17:

"Kotlin is an official language for Android development"

The Java Ecosystem Embraces Kotlin

- Spring Initializr
- Kotlin Compiler Plugin
- Kotlin Support in Spring 5.0

- Kotlin Gradle DSL

- `@TestInstance(Lifecycle.PER_CLASS)`

- Kotlin Android Extensions

Kotlin at GitHub

Fastest growing languages

We're seeing trends toward more statically typed languages focused on thread safety and interoperability: Kotlin, TypeScript, and Rust are growing fast this year.

In addition, the number of contributors writing HCL, a human readable language for DevOps, has more than doubled since 2017. Popular in machine learning projects, Python is at #8. And there are 1.5x more contributors writing Go this year than last year.

		Growth in contributors
1	Kotlin	2.6x
2	HCL	2.2x
3	TypeScript	1.9x
4	PowerShell	1.7x
5	Rust	1.7x
6	CMake	1.6x
7	Go	1.5x
8	Python	1.5x
9	Groovy	1.4x
10	SQLPL	1.4x

https://octoverse.github.com/projects#languages

Pitfalls

Missing Default Constructor for Data Classes

```
data class Snippet(val code: String, val author: String)
val snippet = Snippet()
```

⇒ Issues with Object Mapping:

- JAXB requires default constructor ↯
- Jackson: jackson-module-kotlin allows parameterized constructors
- Hibernate: kotlin-noarg compiler plugin for JPA → Synthetic default constructor

```
apply plugin: "kotlin-jpa"
```

Final by Default

```
class CustomerService {
    fun findCustomer(id: Int){
        //...
    }
}
```

Can’t be extended by subclasses!

- Some frameworks rely on extension of classes
  - Spring
  - Mockito
- Solutions:
  - Interfaces
  - `Open` classes and methods explicitly
  - Spring: '`kotlin-spring`' Open-all-plugin for Kotlin compiler.
  - Mockito: Use MockK instead or Mockito's Incubating Feature

Unit Testing: Change Lifecycle of Test Class

```
@TestInstance(TestInstance.Lifecycle.PER_CLASS)
class MongoDAOTest {
    private val mongo = createMongoContainer().apply {
        configure()
    }
    private val mongoDAO = MongoDAO(mongo.host, mongo.port)

    @Test
    fun foo() {
       // test mongoDAO
    }
}
```

Minor Annoying Aspects of Kotlin

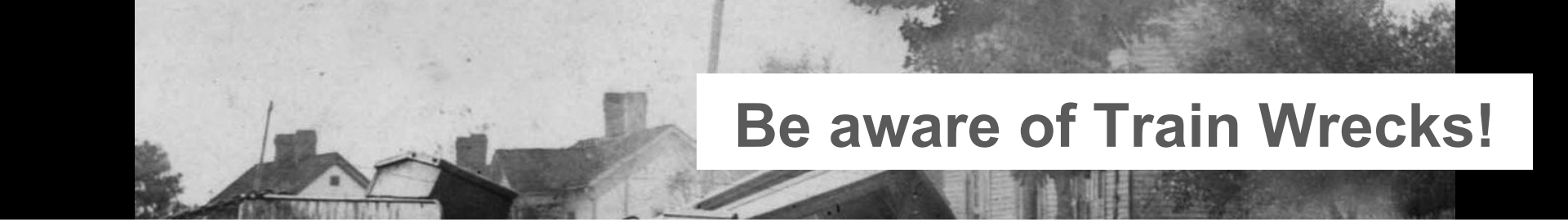
- Language:
  - No package-private visibility
  - No multi-catch

Annoys rarely

- Compilation Speed
  - Build is slower
  - IDE feels slower

Will become better;
Gradle: Inkr. Builds, Daemons

Hammer and Nails



Be aware of Train Wrecks!

```
fun map(dto: OrderDTO, authData: RequestAuthData) = OrderEntity(
    id = dto.id,
    shopId = try {
        extractItemIds(dto.orderItems[0].element.href).shopId
    } catch (e: BatchOrderProcessingException) {
        restExc("Couldn't retrieve shop id from first order item: ${e.msg}")
    },
    batchState = BatchState.RECEIVED,
    orderData = OrderDataEntity(
        orderItems = dto.orderItems.map { dto -> mapToEntity(dto) },
        shippingType = dto.shipping.shippingType.id,
        address = mapToEntity(dto.shipping.address),
        correlationOrderId = dto.correlation?.partner?.orderId,
        externalInvoiceData = dto.externalInvoiceData?.let { ExternalInvoiceDataEntity(
                url = it.url,
                total = it.total,
                currencyId = it.currency.id
        )}
    ),
    partnerUserId = authData.sessionOwnerId ?: restExc("No sessionId supplied", 401),
    apiKey = authData.apiKey,
    dateCreated = if (dto.dateCreated != null) dto.dateCreated else Instant.now(),
)
```

Hammer and Nails

Be careful with:

- Unreadable monster expressions
- Complicated null-safe-calls and elvis structures

```
//Don't
value?.emptyToNull()?.let { map.put("bla", it) }
```

```
// KISS!
if (!value.isNullOrEmpty()){
    map.put("key", value!!)
}
```

```
fun  String.emptyToNull() =
    if (this.isEmpty()) null
    else this
```

Conclusion


Kotlin at Spreadshirt:
A Success Story!

Questions?