Is Boilerplate Code Really So Bad?

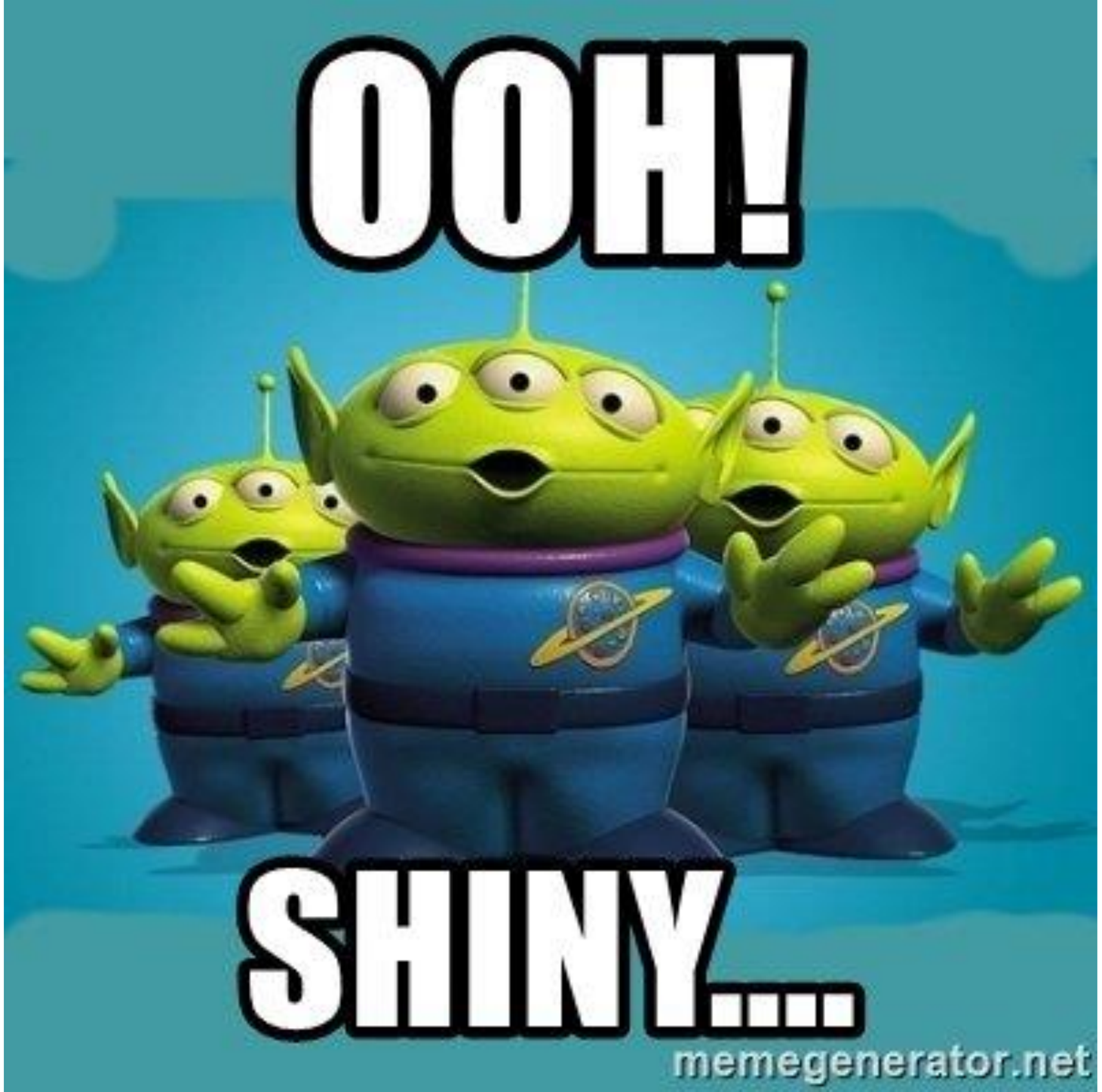
Trisha Gee (@trisha_gee)
Developer & Technical Advocate, JetBrains

TL;DR

Yes

Is Boilerplate Code Really So Bad?

Trisha Gee (@trisha_gee)
Developer & Technical Advocate, JetBrains


OOH!
SHINY....
memegenerator.net


I’m not dead!

```java
private void processEmbeddedAnnotations(final DBCollection dbColl, final MappedClass mc, final boolean background,
                                        final List<MappedClass> parentMCs, final List<MappedField> parentMFs) {
    List<MappedField> annotatedWith = mc.getFieldsAnnotatedWith(Text.class);
    if (annotatedWith.size() > 1) {
        throw new MappingException("Only one text index can be defined per collection: " + mc.getClazz().getName());
    }
    for (final MappedField mf : mc.getPersistenceFields()) {
        if (mf.hasAnnotation(Indexed.class)) {
            final Indexed index = mf.getAnnotation(Indexed.class);
            final StringBuilder prefix = new StringBuilder();
            if (!parentMCs.isEmpty()) {
                for (final MappedField pmf : parentMFs) {
                    prefix.append(pmf.getNameToStore()).append(".");
                }
            }

            final BasicDBObject oldOptions = (BasicDBObject) extractOptions(index);
            final IndexOptions options = index.options();
            final BasicDBObject newOptions = (BasicDBObject) extractOptions(options, false);
            if (!oldOptions.isEmpty() && !newOptions.isEmpty()) {
                throw new MappingException("Mixed usage of deprecated @Indexed value with the new @IndexOption values is not "
                                           + "allowed.  Please migrate all settings to @IndexOptions");
            }
            if (!newOptions.isEmpty()) {
                ensureIndex(dbColl, new BasicDBObject(prefix + mf.getNameToStore(), index.value().toIndexValue()), newOptions);
            } else {
                ensureIndex(dbColl,
                            index.name(),
                            new BasicDBObject(prefix + mf.getNameToStore(), index.value().toIndexValue()),
                            index.unique(),
                            index.dropDups(),
                            index.background() || background,
                            index.sparse(),
                            index.expireAfterSeconds());
            }
        }

        if (mf.hasAnnotation(Text.class)) {
            createTextIndex(dbColl, parentMCs, parentMFs, mf);
        }

        if (!mf.isTypeMongoCompatible() && !mf.hasAnnotation(Reference.class) && !mf.hasAnnotation(Serialized.class)
                && !mf.hasAnnotation(NotSaved.class) && !mf.hasAnnotation(Transient.class)) {
            final List<MappedClass> newParentClasses = new ArrayList<MappedClass>(parentMCs);
            final List<MappedField> newParents = new ArrayList<MappedField>(parentMFs);
            newParentClasses.add(mc);
            newParents.add(mf);
            ensureIndexes(dbColl, mapper.getMappedClass(mf.isSingleValue() ? mf.getType() : mf.getSubClass()), background,
                          newParentClasses, newParents);
        }
    }
}
```

My IDE Will Save Me

Being expressive is better than being terse?

Modern languages reduce unnecessary syntax

You can focus on the business logic

Java is evolving

Symbiotic relationship with languages like Kotlin

Show me some code!

Hello World

Declaring Variables

Java 5

```
Map<Integer, Customer> customers1 = new HashMap<Integer, Customer>();
```

Java 5 + IDE Support

```
Map<Integer, Customer> customers1 = new HashMap<~>();
```

Java 7

```
Map<Integer, Customer> customers2 = new HashMap<>();
```

Kotlin

```
var customers1 = HashMap<Int, Customer>()
```

Java 10

```
var customers4 = new HashMap<>();ger, Customer>();
```

Data Classes

http://cr.openjdk.java.net/~briangoetz/amber/datum.html

Casting

Java

```java
void createSegment(Object obj) {
    if (obj instanceof View) {
        ((View) obj).initialise();
    }
}
```

Kotlin

```kotlin
fun createSegment(obj: Any?) {
    if (obj is View) {
        obj.initialise()
    }
}
```

Nulls

Java

```
void validateCustomer(CustomerJava customer) {
    if (customer ≠ null) {
        if (customer.getName() ≠ null) {
            if (customer.getName().startsWith("A")) {
                throw new SecurityException("Names are not allowed to begin with A");
            }
        }
    }
}
```

Java 8

```
void validateCustomer(Optional<CustomerOptional> customer) {
    customer.flatMap(CustomerOptional::getName)
            .filter(name → name.startsWith("A"))
            .ifPresent(s → throwSecurityException("Names are not allowed to begin wi
}
```

Kotlin

```
fun validateCustomerWithNulls(customer: CustomerJava?) {
    if (customer?.name?.startsWith("A") == true) {
        throw Exception("Names are not allowed to start with A")
    }
}
```

Switch

```
int port = Integer.valueOf(portInputValue);
PortType type = PortType.UNKNOWN;
switch (port) {
    case 20:
        type = PortType.FTP;
        break;
    case 80:
        type = PortType.HTTP;
        break;
    case 8080:
        type = PortType.HTTP;
        break;
    case 27017:
        type = PortType.DATABASE;
        break;
    default:
        if (port ⩾ 20_001 && port ⩽ 30_000) {
            type = PortType.SAFE;
        } else if (port ⩾ 9080 && port ⩽ 9092) {
            type = PortType.BUSY;
        }
}
```

```
val type = when (port) {
    20 → PortType.FTP
    80, 8080 → PortType.HTTP
    27017 → PortType.DATABASE
    in 20001..30000 → PortType.SAFE
    in 9080..9092 → PortType.BUSY
    !is Int → throw RuntimeException("Not a valid port number")
    else → PortType.UNKNOWN
}
```

```
int port = Integer.valueOf(portInputValue);
PortType type = PortType.UNKNOWN;
switch (port) {
    case 20:
        type = PortType.FTP;
        break;
    case 80:
        type = PortType.HTTP;
        break;
    case 8080:
        type = PortType.HTTP;
        break;
    case 27017:
        type = PortType.DATABASE;
        break;
    default:
        if (port ⩾ 20_001 && port ⩽ 30_000) {
            type = PortType.SAFE;
        } else if (port ⩾ 9080 && port ⩽ 9092) {
            type = PortType.BUSY;
        }
}
```

```
val type = when (port) {
    20 → PortType.FTP
    80, 8080 → PortType.HTTP
    27017 → PortType.DATABASE
    in 20001..30000 → PortType.SAFE
    in 9080..9092 → PortType.BUSY
    !is Int → throw RuntimeException("Not a valid port number")
    else → PortType.UNKNOWN
}
```

Default parameter values

Java

```java
void printMessage(String message) {
    printMessage(message, "", "");
}
void printMessage(String message, String prefix) {
    printMessage(prefix, message, "");
}
void printMessage(String message, String prefix, String suffix) {
    System.out.format("%s %s %s", message, prefix, suffix);
}
```

Kotlin

```kotlin
fun printMessage(message: String, prefix: String = "", suffix: String = "") {
    println("$prefix $message $suffix")
}
```

Java

```java
void printMessage(String... messages) {
    for (String message : messages) {
        System.out.printf("%s ", message);
    }
}
```

Ranges

Java 8

```
IntStream numbers = IntStream.range(1, 100);
```

Kotlin

```
val numbers = 1..100
```

Java 10

```
var numbers = range(1, 100);
```

Collections

Java

```
List<CustomerJava> customers1 = Arrays.asList(st(asList(
        new CustomerJava(1, "Sam", "Sparks"),
        new CustomerJava(2, "Pat", "Parks"));
```

Java 9

```
List<CustomerJava> customers3 = List.of(
        new CustomerJava(1, "Sam", "Sparks"),
        new CustomerJava(2, "Pat", "Parks"));
```

Kotlin

```
val customers = listOf(
        CustomerKotlin(1, "Sam", "Sparks"),
        CustomerKotlin(2, "Pat", "Parks"))
```

Java 10

```
var customers4 = of(
        new CustomerJava(1, "Sam", "Sparks"),
        new CustomerJava(2, "Pat", "Parks"));
```

Lambda Expressions

Java

```
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(final ActionEvent e) {
        System.out.println("I was pushed, I didn't fall!");
    }
});
```

Java 8

```
button.addActionListener(event → System.out.println("I was pushed, I didn't
```

Kotlin

```
button.addActionListener { println("I was pushed, I didn't fall!") }
```

Java ??

```
button.addActionListener(_ → System.out.println("I was pushed, I didn't fal
```

Functional Parameters

Java 8

```
public static void main(String[] args) {
    get("/home", (request, response) → { /*do something */ } );
}

private static void get(final String path, final RouteHandlerJava handler) {
    handler.handle(request, response);
}

@FunctionalInterface
public interface RouteHandlerJava {
    void handle(RequestJava request, ResponseJava responseJava);
}
```

Java 8

```
public static void main(String[] args) {
    get("/home", (request, response) → { /*do something */ } );
}

private static void get(String path, BiConsumer<RequestJava, ResponseJava> handler) {
    handler.accept(request, response);
}
```

Kotlin

```kotlin
fun main(args: Array<String>) {
    get("/home") { request, response → /*do something */ }
}

fun get(path: String, handler: (RequestKotlin, ResponseKotlin) → Unit) {
    handler(request, response)
}
```

Navigating Collections

Java 8

```
List<String> emails = customers4.stream()
                        .filter(customer → customer.getName().startsWith("A"))
                        .map(CustomerJava::getEmail)
                        .collect(Collectors.toList());
```

Kotlin

```
val emails = customers
                .filter { it.name.startsWith("A") }.startsWith("A") }
                .map { it.email }
```

Kotlin

```
val emails = customers.asSequence()
                .filter { it.name.startsWith("A") }
                .map { it.email }
                .toList()
```

In Summary

Boilerplate can obscure business logic

Modern languages remove boilerplate

Readability is objective


JET
BRAINS
http://bit.ly/BoilJVM