OpenRewrite in a Nutshell

Scaling Upgrades with Practical Insights

Simon Gartner

01.07.2025

gepardec
simplify your business

Frameworks we've used

Java™ Enterprise Edition

spring® 3

QUARKUS 2

JUnit

LOG4J 1

Upgraded to...

JAKARTA® EE

spring® 6

QUARKUS 3

JUnit 5

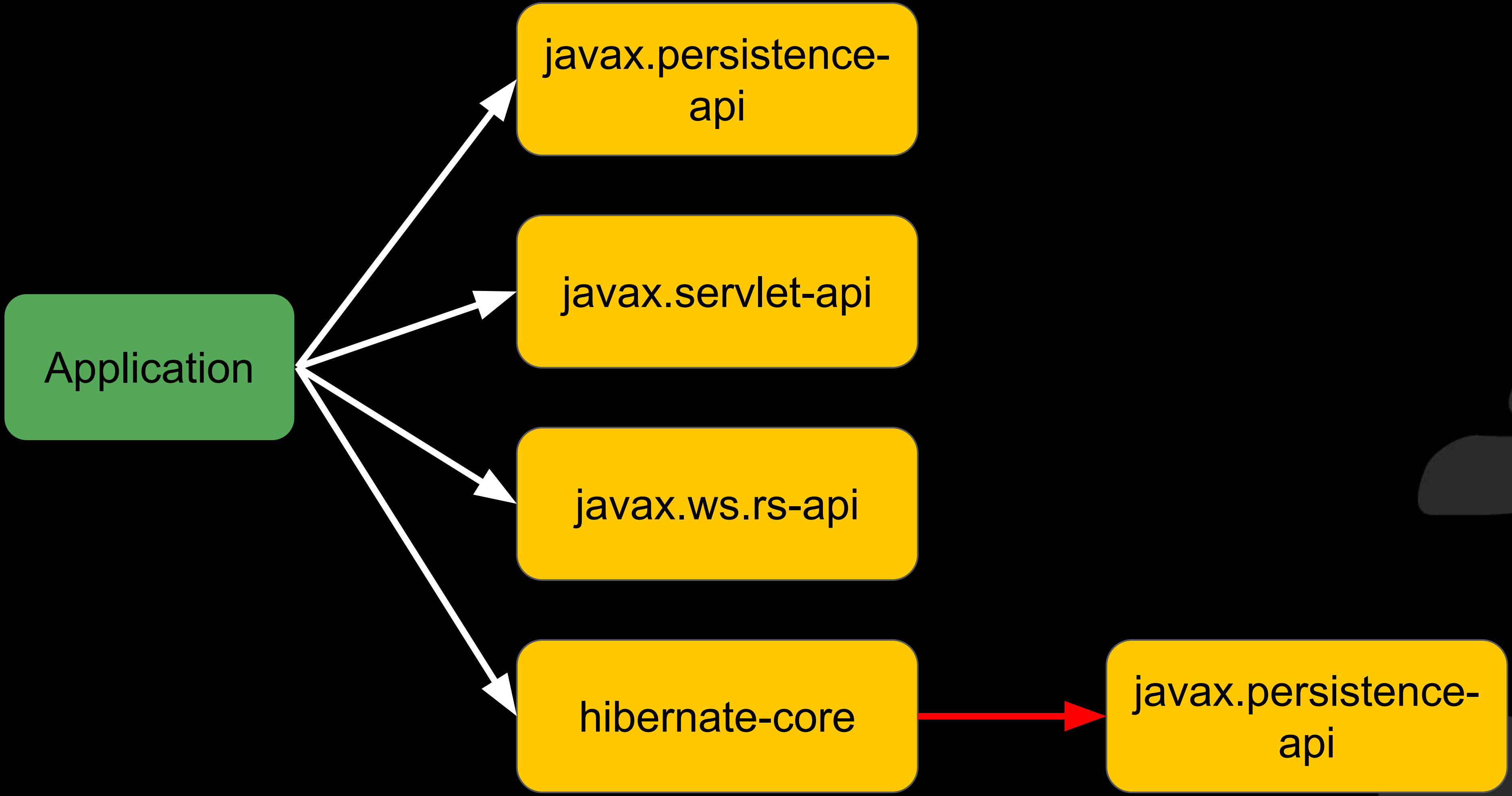
LOG4J 2

A large upgrade: Jakarta EE 9

// **javax**.inject.Inject -> **jakarta**.inject.Inject

// **javax**.naming.InitialContext -> **javax**.naming.InitialContext

A large upgrade: Jakarta EE 9



gepardec
simplify your business

Upgrading manually - A tedious task

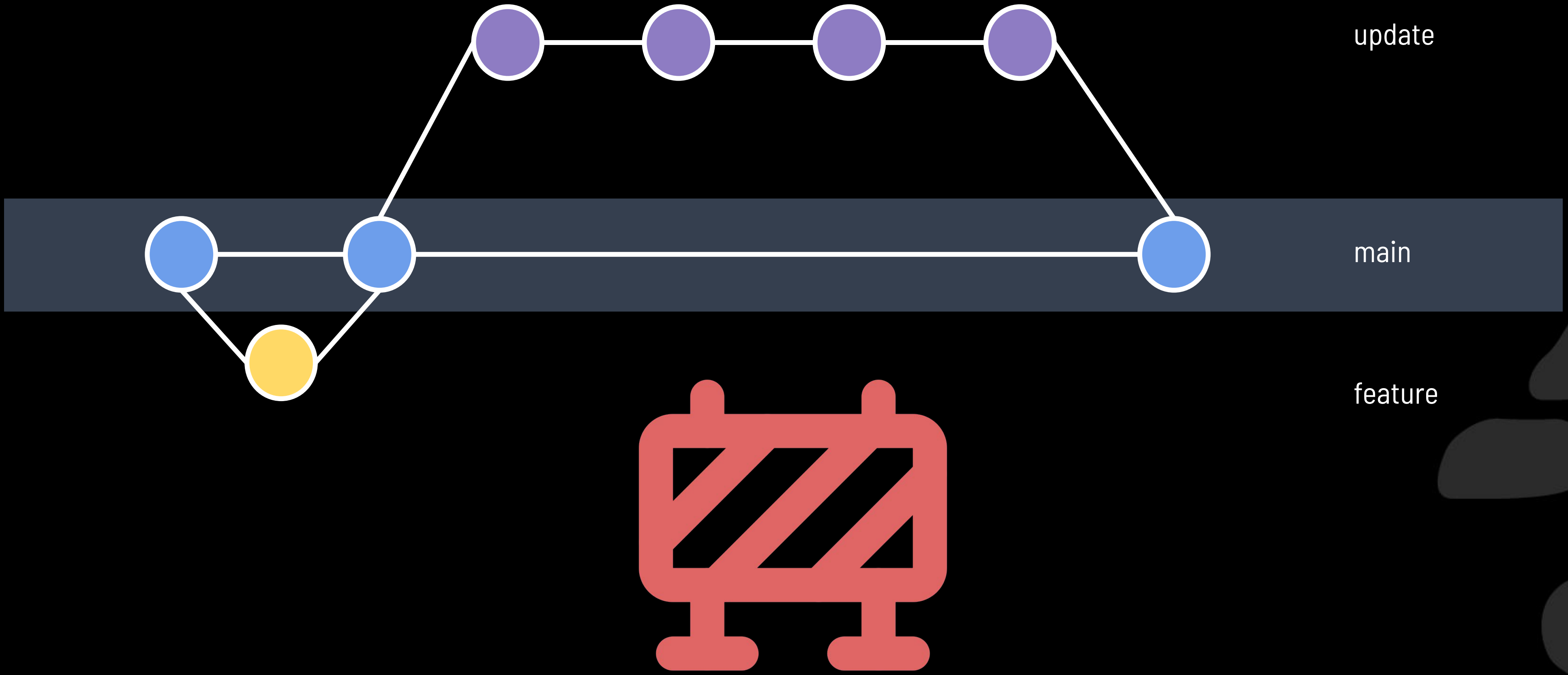
// Time consuming

// Error prone

// Not fun

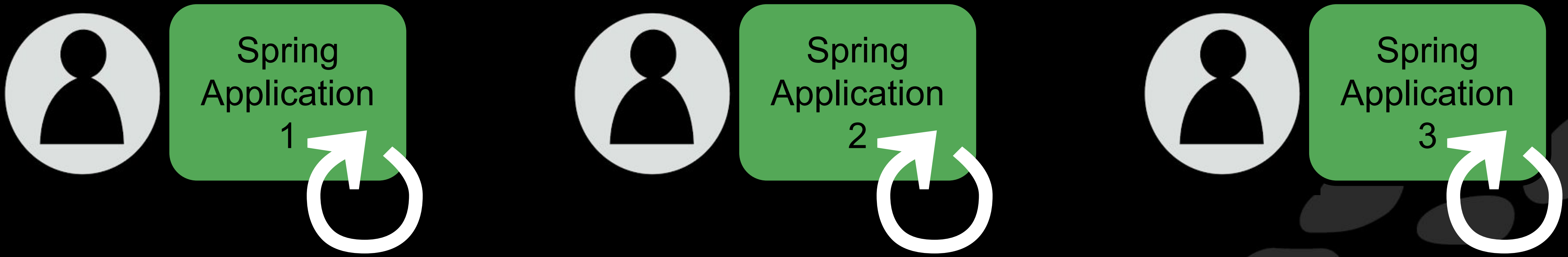
gepardec
simplify your business

Upgrading manually - Stop the world

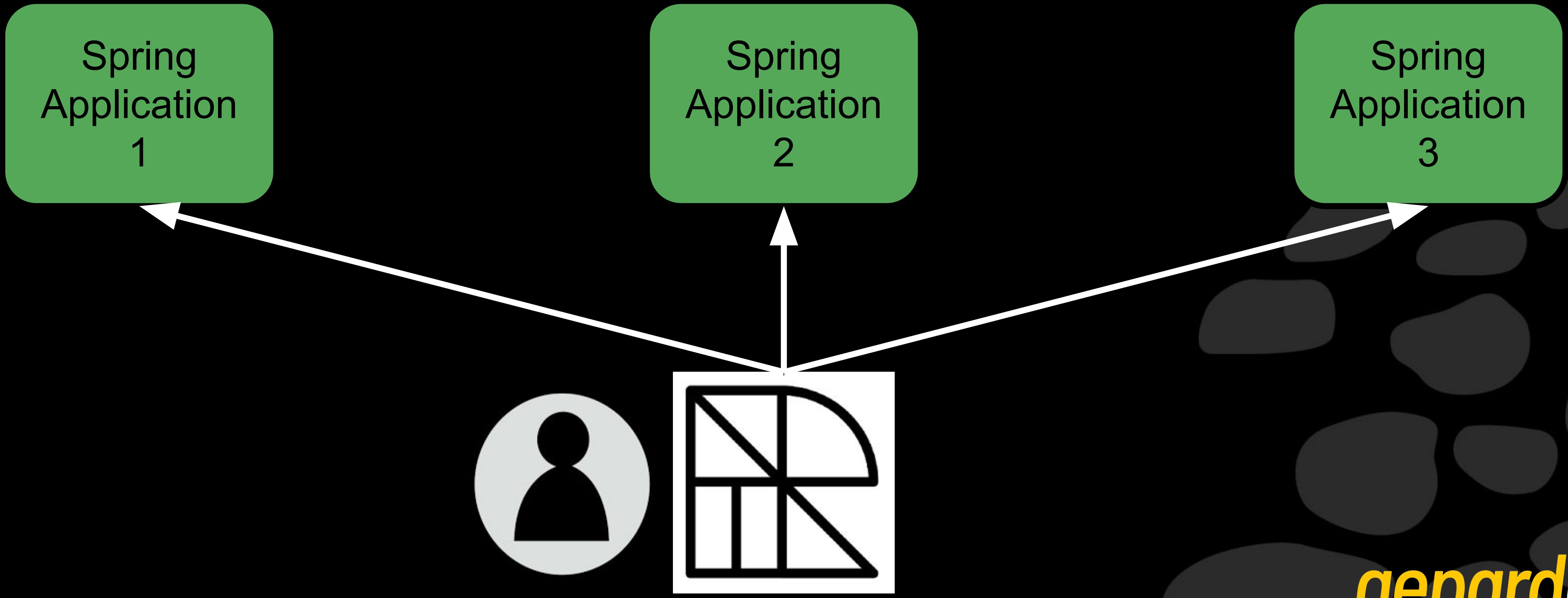


gepardec
simplify your business

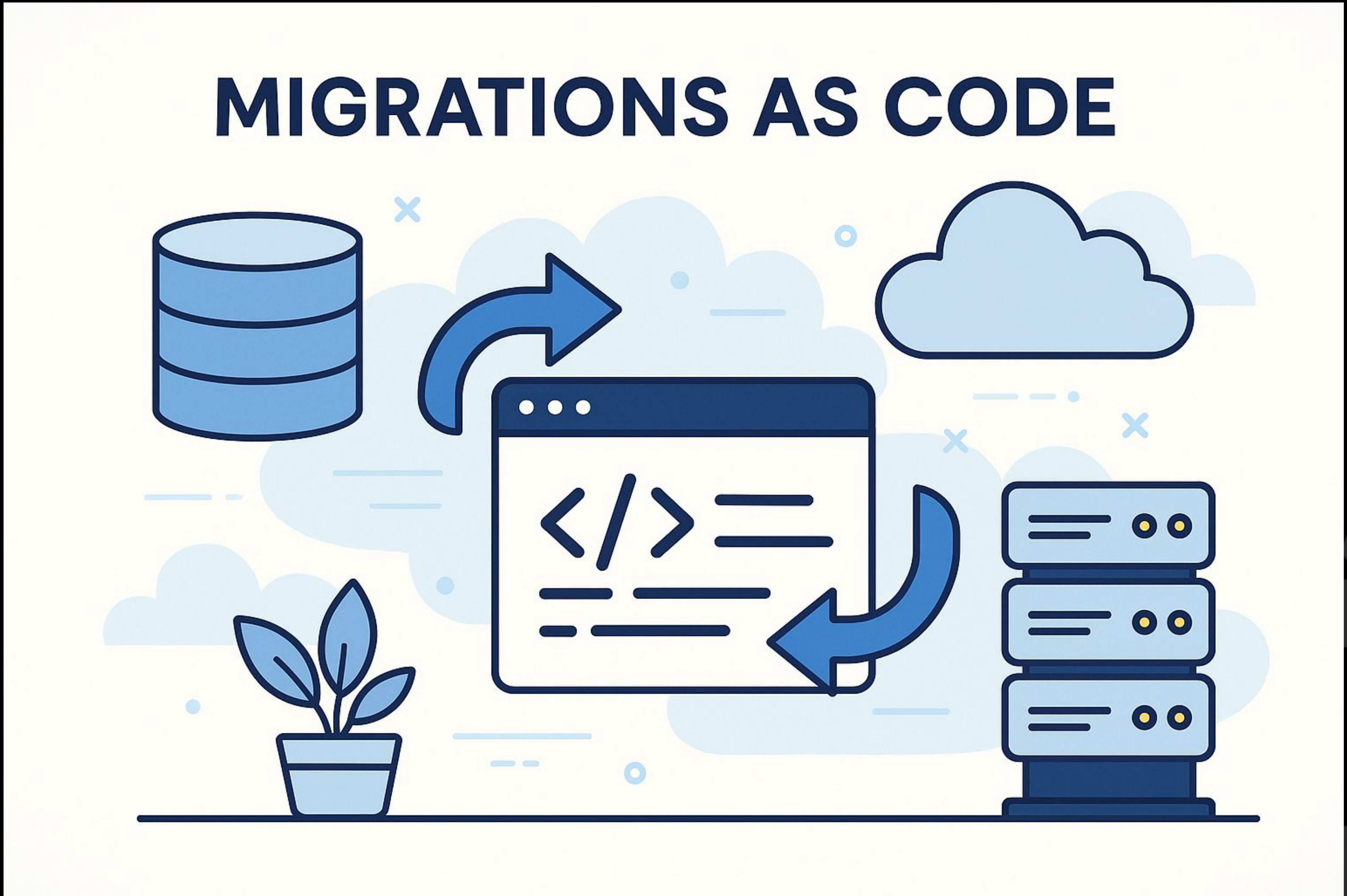
Upgrading manually - Repeated effort



And that's how OpenRewrite got born!



gepardec
simplify your business


MIGRATIONS AS CODE

Formats supported by OpenRewrite

// Java

// XML / Maven

// Groovy / Gradle

// JSON

// YAML

// Properties

// ...

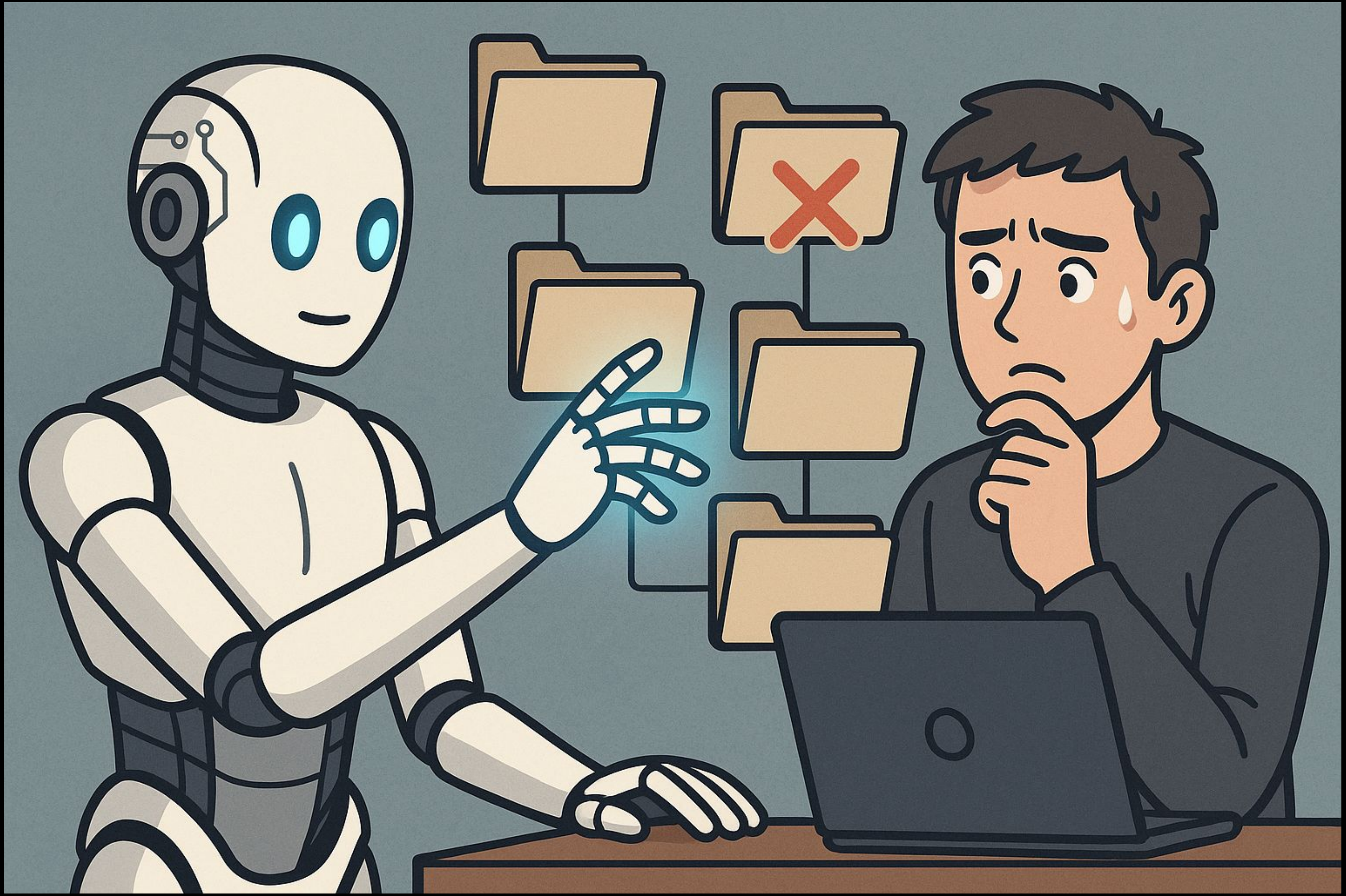
gepardec
simplify your business

About text replacements

```
public void example() {
  Type x = new Type();
  x.doSomething();
  OtherType y = new OtherType();
  y.doSomething();
}
```

```
public void example() {
  Type x = new Type();
  x.doNothing();
  OtherType y = new OtherType();
  y.doSomething();
}
```

About LLMs



gepardec
simplify your business

Recipe Catalog

// List of official Recipes by Moderne / Open Source

https://docs.openrewrite.org/recipes/java/migrate/jakarta/jakartaee11

Composite Recipes

gepardec

simplify your business

Another Recipe

https://docs.openrewrite.org/recipes/maven/changedependencygroupidandartifactid

gepardec
simplify your business

Other Recipes

// UpgradeDependency

// ChangePropertyValue

// ChangeType

// ReplaceAnnotation

// JakartaEE10

// Slf4jToLog4j

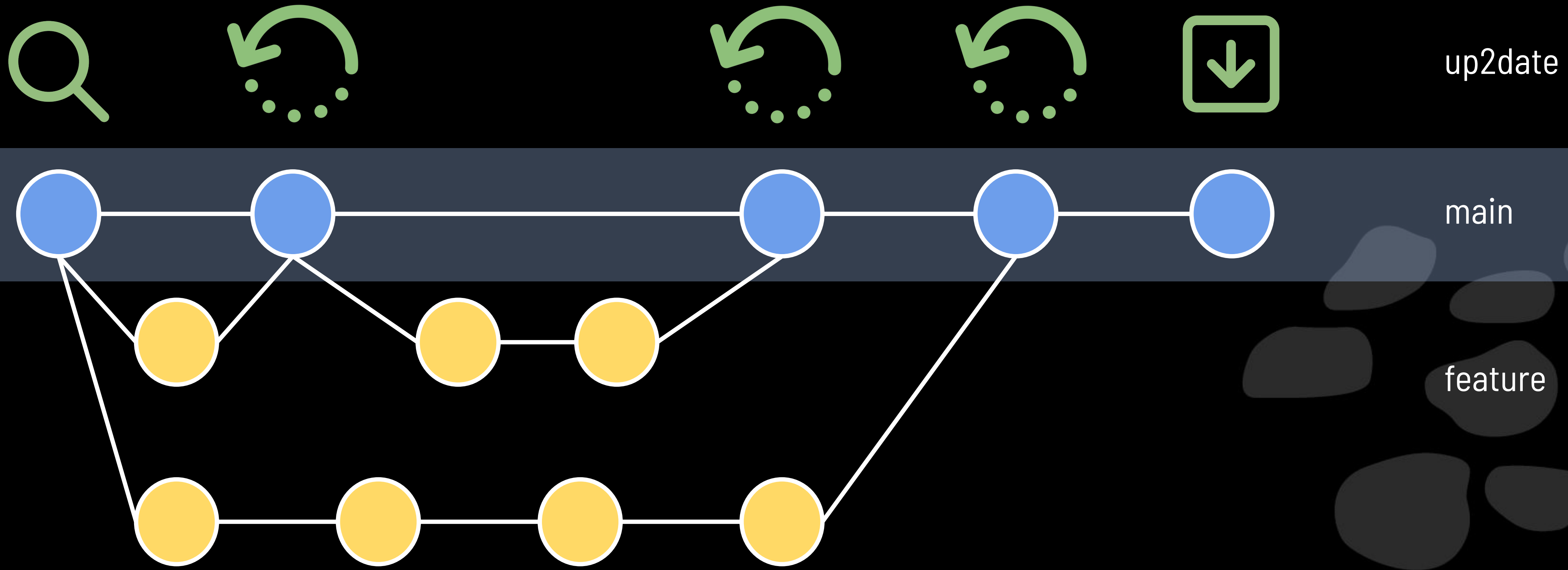
gepardec
simplify your business

Framework Migrations

gepardec

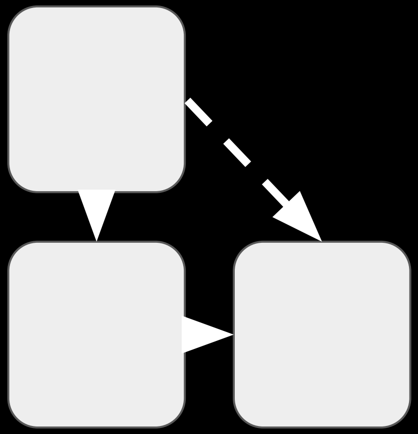
simplify your business

Transformer



gepardec
simplify your business

Prerequisites for Framework Migrations



// Transitive dependencies in controlled code?

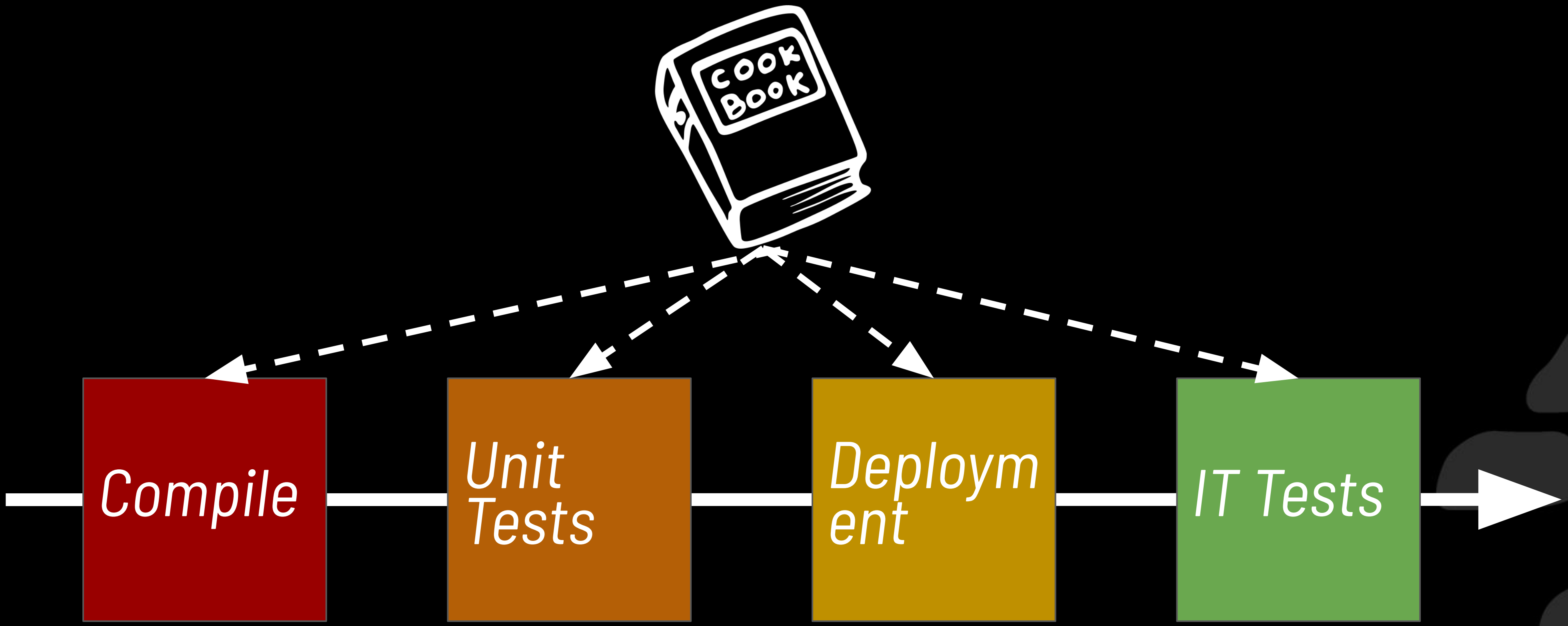
// Ensure a good test coverage

// Clean up to ease automation

gepardec

simplify your business

Automating Framework Migrations



gepardec
simplify your business

Single Cases

Q: Should I make the effort to integrate specific cases even if they only have one occurrence?

A: Yes! But keep it simple! For example, you could do:

- FindAndReplace
- Replace entire files

Custom Recipes

gepardec

simplify your business

Interface transformation

```
request
   .getMetadata()
   .setTenantId(“1”);
```

```
Metadata meta =
ObjectFactory.createMetadata();
JaxbElement<String> tenantId =
   ObjectFactory
     .createMetadataTenantId(“1”);
meta.setTenantId(tenantId);
request.setMetadata(meta);
```

Lossless Semantics Tree (LST)

```
\—-J.ClassDeclaration
    |—-J.Modifier | "public"
    |—-J.Identifier | "Main"
     \—-J.Block
         \———-J.MethodDeclaration | "MethodDeclaration{com.gepardec.Main}"
                 |—-J.Modifier | "public"
                 |—-J.Modifier | "static"
                 |—-J.Primitive | "void"
                 |—-J.Identifier | "main"
                 |——————-J.VariableDeclarations | "String[ ] args"
                  \—-J.Block
                      \———-J.MethodInvocation | "System.out.println("Hello world!")"
                          |———-J.FieldAccess | "System.out"
                          |—-J.Identifier | "println"
                           \——————-J.Literal | ""Hello world!""
```

public class Main {...}

Lossless Semantics Tree (LST)

```
\—-J.ClassDeclaration
   |—-J.Modifier | "public"
   |—-J.Identifier | "Main"
    \—-J.Block
       \———-J.MethodDeclaration | "MethodDeclaration{com.gepardec.Main}"
             |—-J.Modifier | "public"
             |—-J.Modifier | "static"
             |—-J.Primitive | "void"
             |—-J.Identifier | "main"
             |——————-J.VariableDeclarations | "String[ ] args"
             \—-J.Block
                 \———-J.MethodInvocation | "System.out.println("Hello world!")"
                       |———-J.FieldAccess | "System.out"
                       |—-J.Identifier | "println"
                       \———————-J.Literal | ""Hello world!""
```

```
public static void main(String[] args)
{...}
```

gepardec
simplify your business

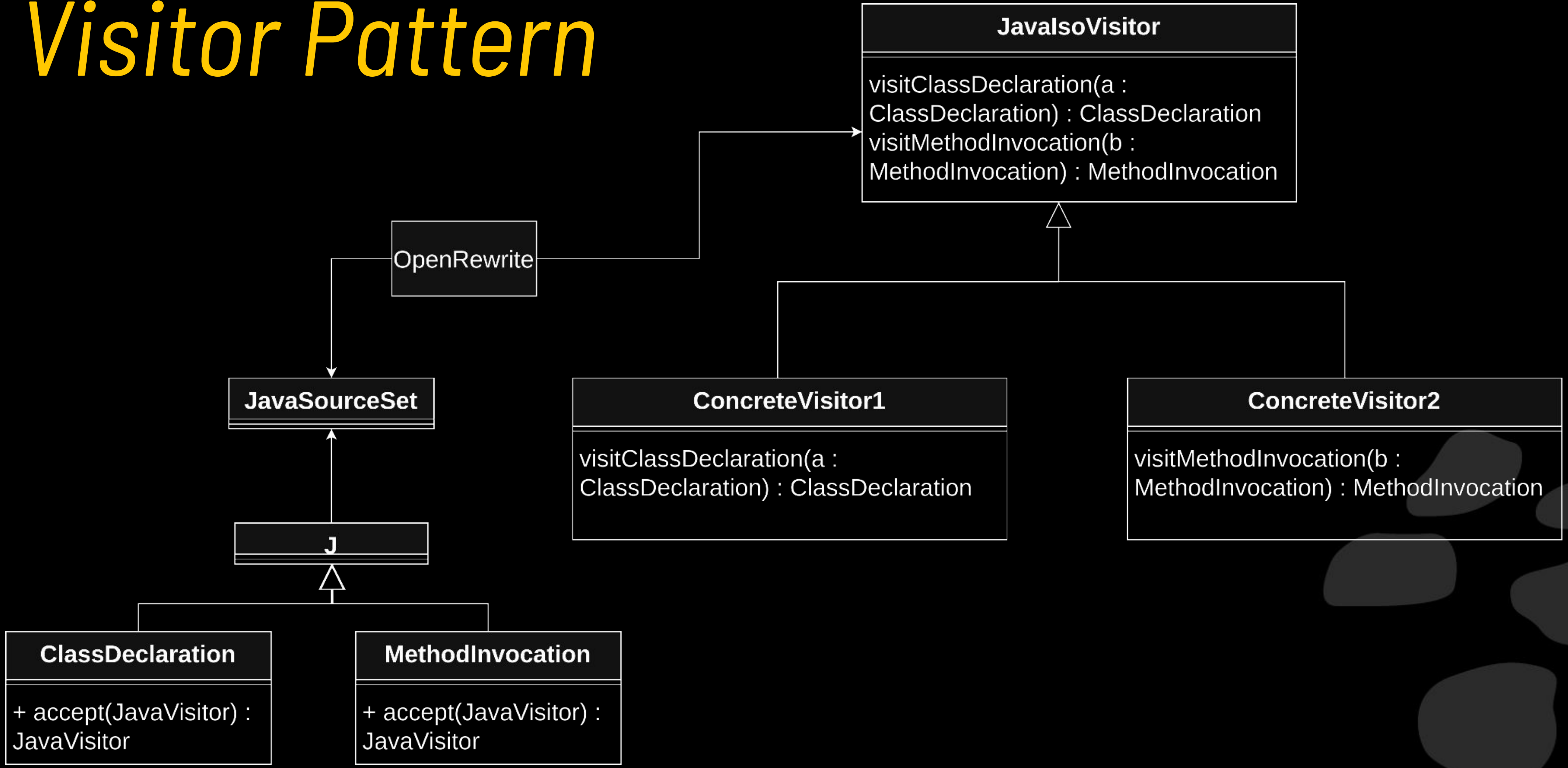
Lossless Semantics Tree (LST)

System.out.println(“Hello world!”);

```
\—-J.ClassDeclaration
   |—-J.Modifier | "public"
   |—-J.Identifier | "Main"
   \—-J.Block
       \———-J.MethodDeclaration | "MethodDeclaration{com.gepardec.Main}"
             |—-J.Modifier | "public"
             |—-J.Modifier | "static"
             |—-J.Primitive | "void"
             |—-J.Identifier | "main"
             |——————-J.VariableDeclarations | "String[ ] args"
             \—-J.Block
                 \———-J.MethodInvocation | "System.out.println("Hello world!")"
                       |———-J.FieldAccess | "System.out"
                       |—-J.Identifier | "println"
                       \———————-J.Literal | ""Hello world!""
```

gepardec
simplify your business

Visitor Pattern


JavaIsoVisitor
visitClassDeclaration(a : ClassDeclaration) : ClassDeclaration
visitMethodInvocation(b : MethodInvocation) : MethodInvocation
OpenRewrite
JavaSourceSet
J
ClassDeclaration
+ accept(JavaVisitor) : JavaVisitor
MethodInvocation
+ accept(JavaVisitor) : JavaVisitor
ConcreteVisitor1
visitClassDeclaration(a : ClassDeclaration) : ClassDeclaration
ConcreteVisitor2
visitMethodInvocation(b : MethodInvocation) : MethodInvocation

gepardec
simplify your business

Example for custom recipes

gepardec

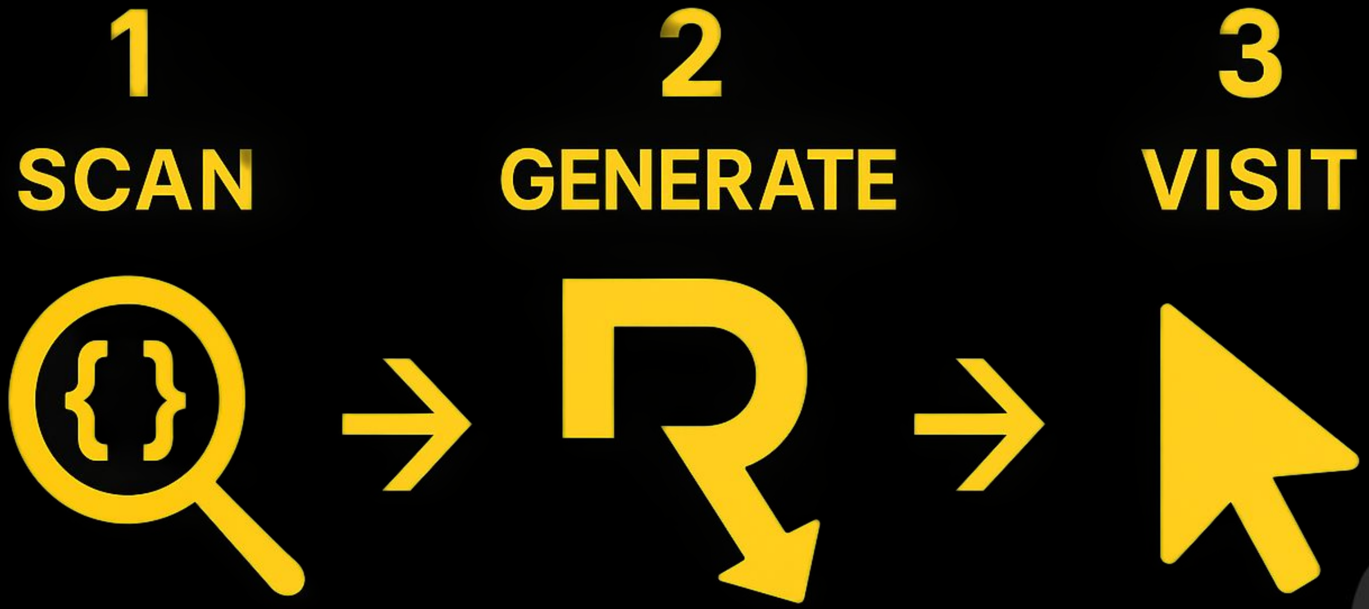
simplify your business

Interface transformation

```
request
   .getMetadata()
   .setTenantId(“1”);
```

```
Metadata meta =
ObjectFactory.createMetadata();
JaxbElement<String> tenantId =
   ObjectFactory
     .createMetadataTenantId(“1”);
meta.setTenantId(tenantId);
request.setMetadata(meta);
```

Scanning Recipe



// Scan DTO hierarchies and store as Tree

// Lookup DTO in tree and add initialisations + nested setters

Quality Criterias for Recipes

// Type attribution

// Idempotency

// Atomicity

Learning to write Recipes

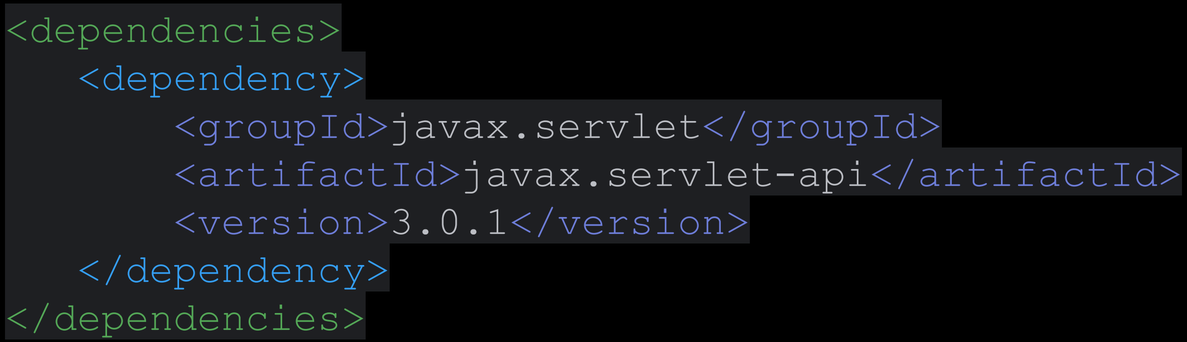
// https://docs.openrewrite.org/authoring-recipes

// https://docs.moderne.io/user-documentation/community-office-hours/

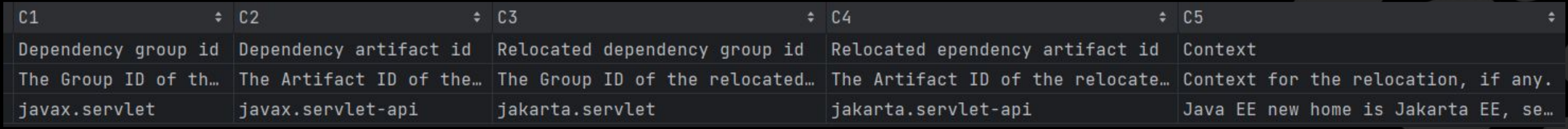
// https://app.moderne.io/recipes/org.openrewrite.java.search.FindMethods

// https://join.slack.com/t/rewriteoss/shared_invite/zt-1ihfggp2a-gllit_aXJnhHAdv_0uzwow

DataTables: RelocatedDependencyCheck



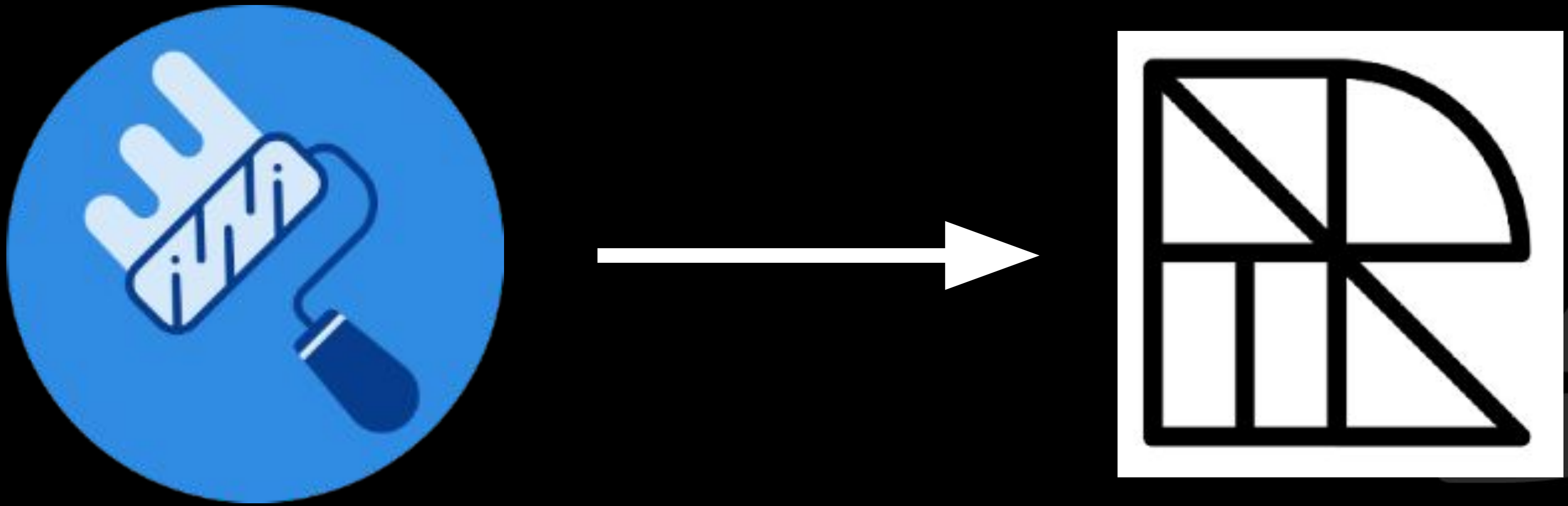
```
<dependencies>
   <dependency>
       <groupId>javax.servlet</groupId>
       <artifactId>javax.servlet-api</artifactId>
       <version>3.0.1</version>
   </dependency>
</dependencies>
```



C1	C2	C3	C4	C5
Dependency group id	Dependency artifact id	Relocated dependency group id	Relocated ependency artifact id	Context
The Group ID of th…	The Artifact ID of the…	The Group ID of the relocated…	The Artifact ID of the relocate…	Context for the relocation, if any.
javax.servlet	javax.servlet-api	jakarta.servlet	jakarta.servlet-api	Java EE new home is Jakarta EE, se…

gepardec
simplify your business

Auto Update Service



// Renovate finds new Updates

// OpenRewrite migrates to a new update

gepardec

simplify your business

TQRCG