

arnaud.nauwynck@gmail.com

# Architecture Design

## Part 3 : DTO

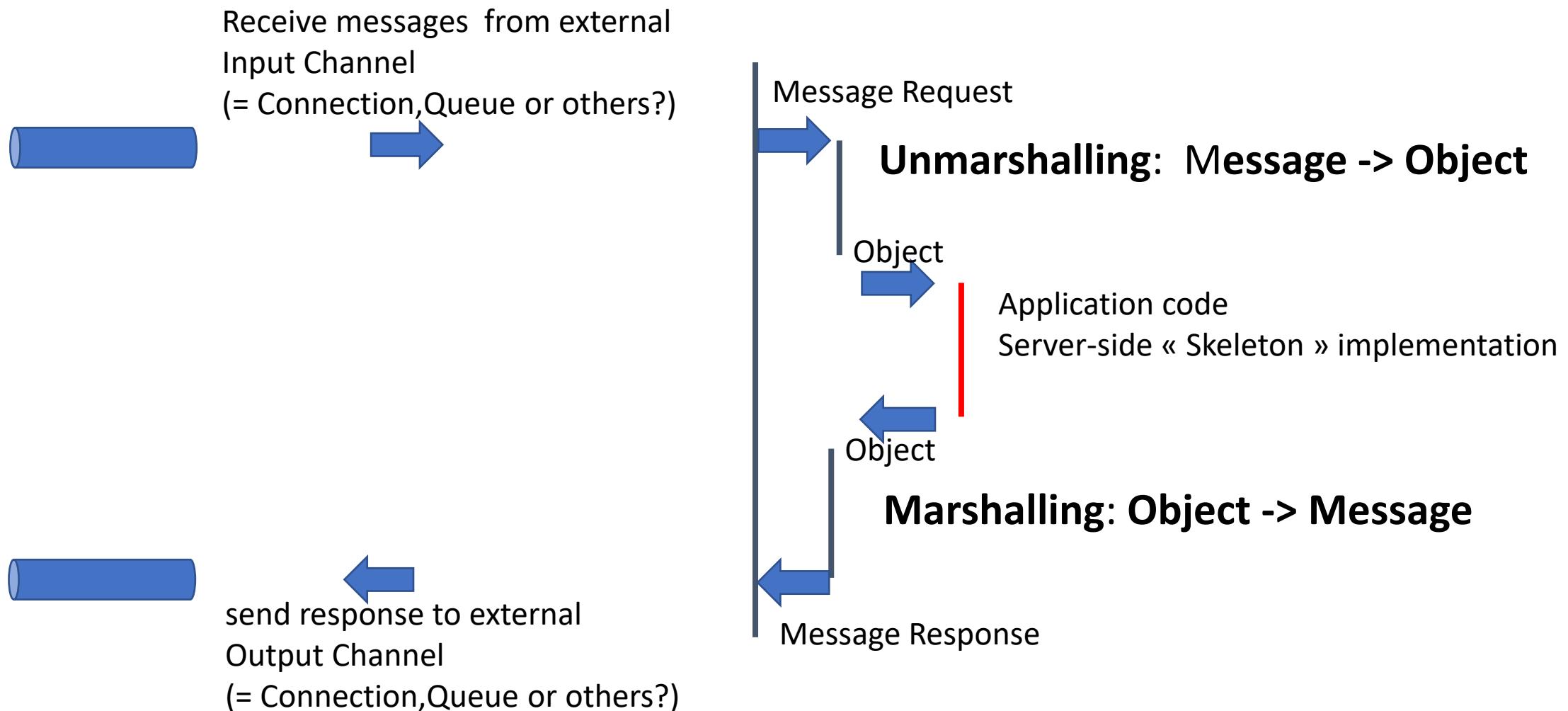
Entity – DTO, external APIs, Protocol Marshalling

This document:

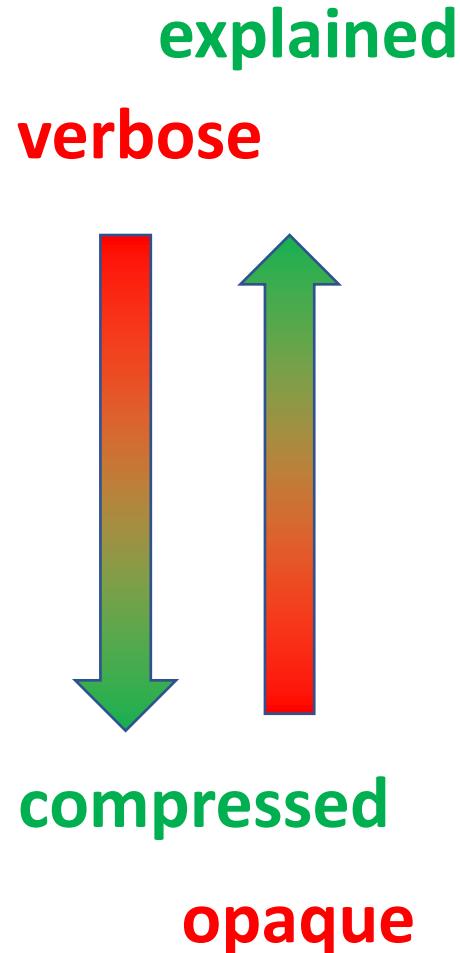
[http://github.com/arnaud-nauwynck/Presentations/java/  
Architecture-Design-part3-DTO.pdf](http://github.com/arnaud-nauwynck/Presentations/java/Architecture-Design-part3-DTO.pdf)

# External Protocol

## ... Marshalling/Unmarshalling to Internal Langage



# Message Encoding : Text / Binary



```
<xml? schema:=« ..xsd » namespace=« ns1:..... » >  
    <a><ns1:b> some value</ns1:b> </a>
```

+++ most powerfull  
+++ Self-explained / extensible  
--- most verbose

```
<xml?>  <a> <b> some value</b> </a>
```

```
JSON: { « a »: { « b »: « some value » } }
```

Simplest  
Compromise for Web

```
CSV: a.b; \n  
      some value
```

Obscure  
+++ compressed encoding  
+++ efficient int32, long64, float32, ..  
CPU representation

```
Binary: 010101010110101010101010
```

# Schema ... Code-Generator

## Text (No code generator)

<?xml> ← xsd schema

{ json} ← ?? No standard  
( json-schema,OpenApi,Swagger)

CSV ← Header line ... Tabular, Not tree

## Binary + code generator per-langage

Corba,RPC,XRD,... ← .idl

Avro ← .avsc : Avro-Schema (in json)

Thrift ← .idl : Thrift Schema

Protobuf,gRPC ← .proto : Protobuf Schema

# Schema: Fixed / support version upgrade

Backward-compatibility is mandatory

Still evolutivity possible ? Better if true



# Dynamic Schema: Protobuf, gRPC compiled Versioned Fields

In schema, all field are numbered (unique ID)

```
message HelloRequest {
    string name = 1;
}

message HelloRequest {
    string name = 1;
    string color = 2; // added in version 2
}

message HelloRequest {
    string name = 1;
    string color = 2; // deprecated, unused in version >=3 .. replaced by cssStyle
    string cssStyle = 3;
}
```

# Dynamic Messages, Schema-compatible

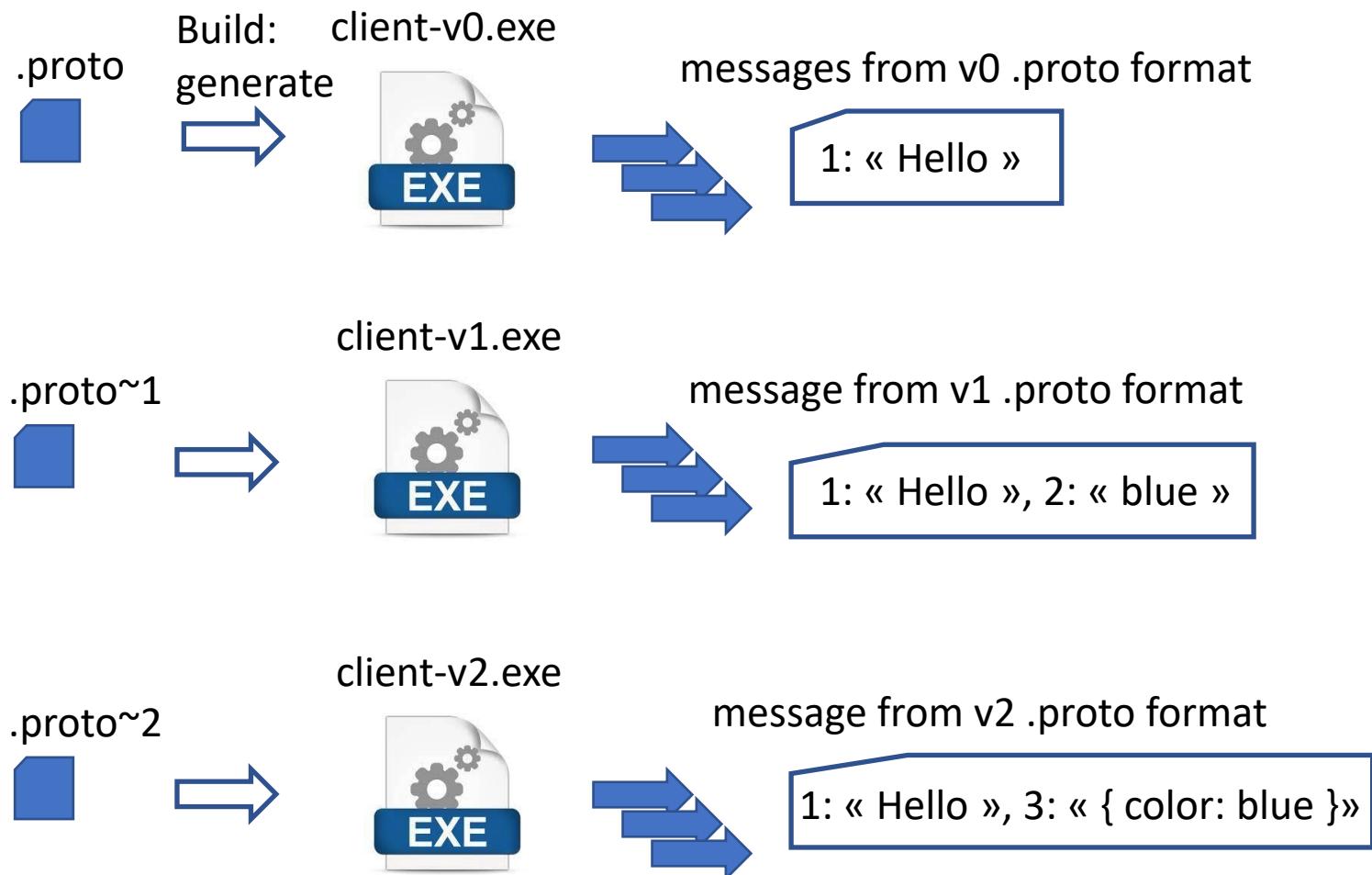
In Messages

« Map<FieldVersion, Value> »

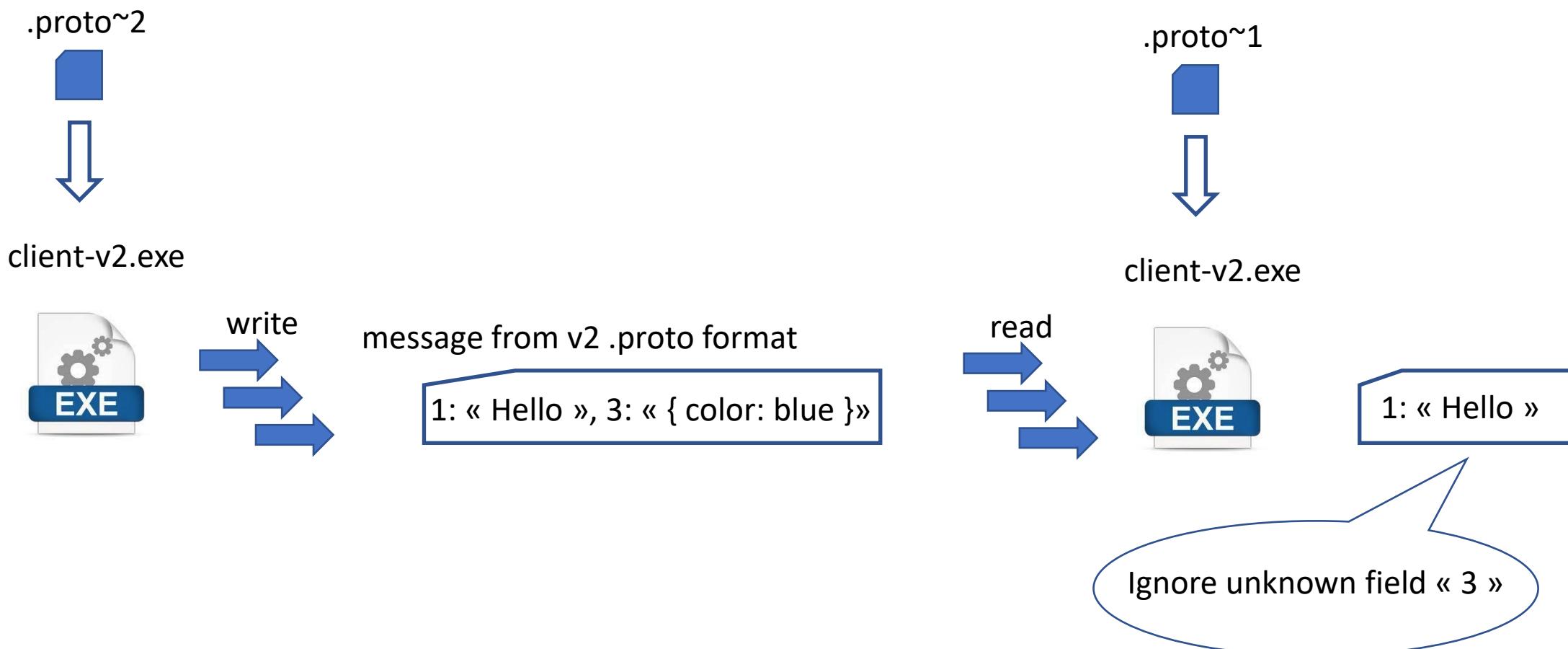
all Field ids + values encoded

Small extra cost

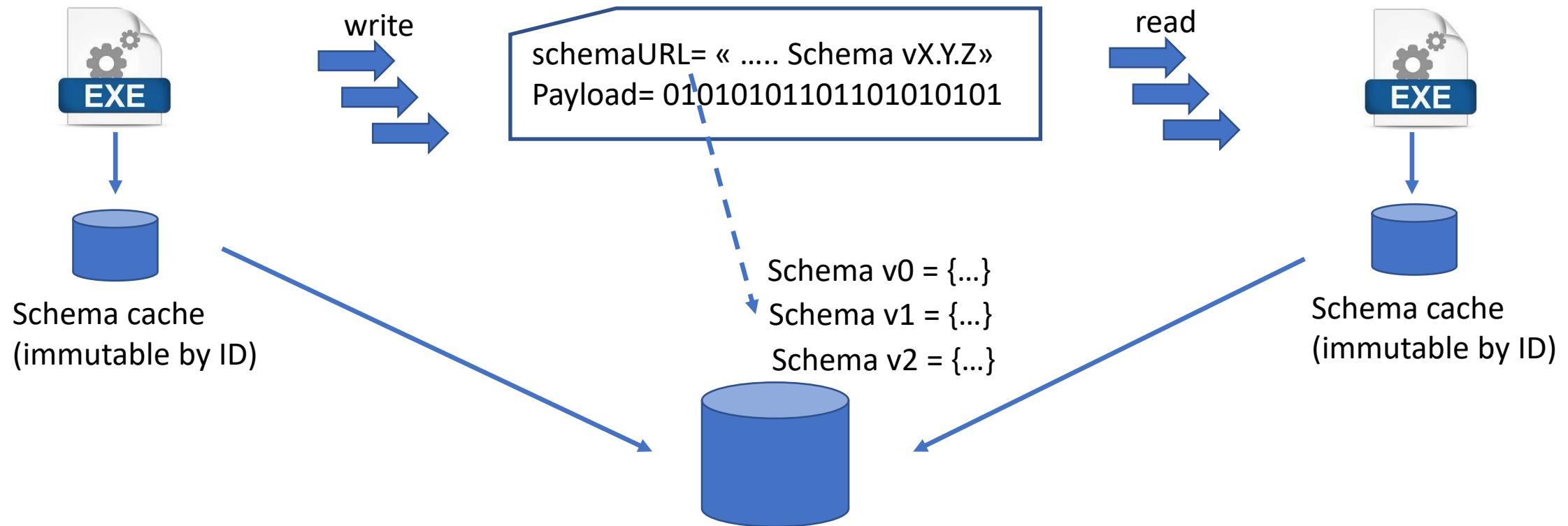
... Great compatibility



# Reading « Future » Message / Backward Client ... Ignore Unknown Fields



# Schema Registry



# Schema Contained in Messages

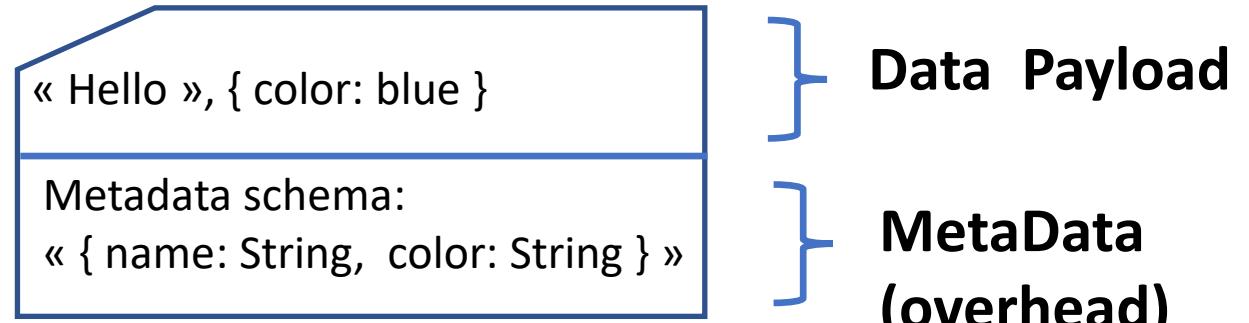
Examples :

Kafka Messages, Pulsar Messages

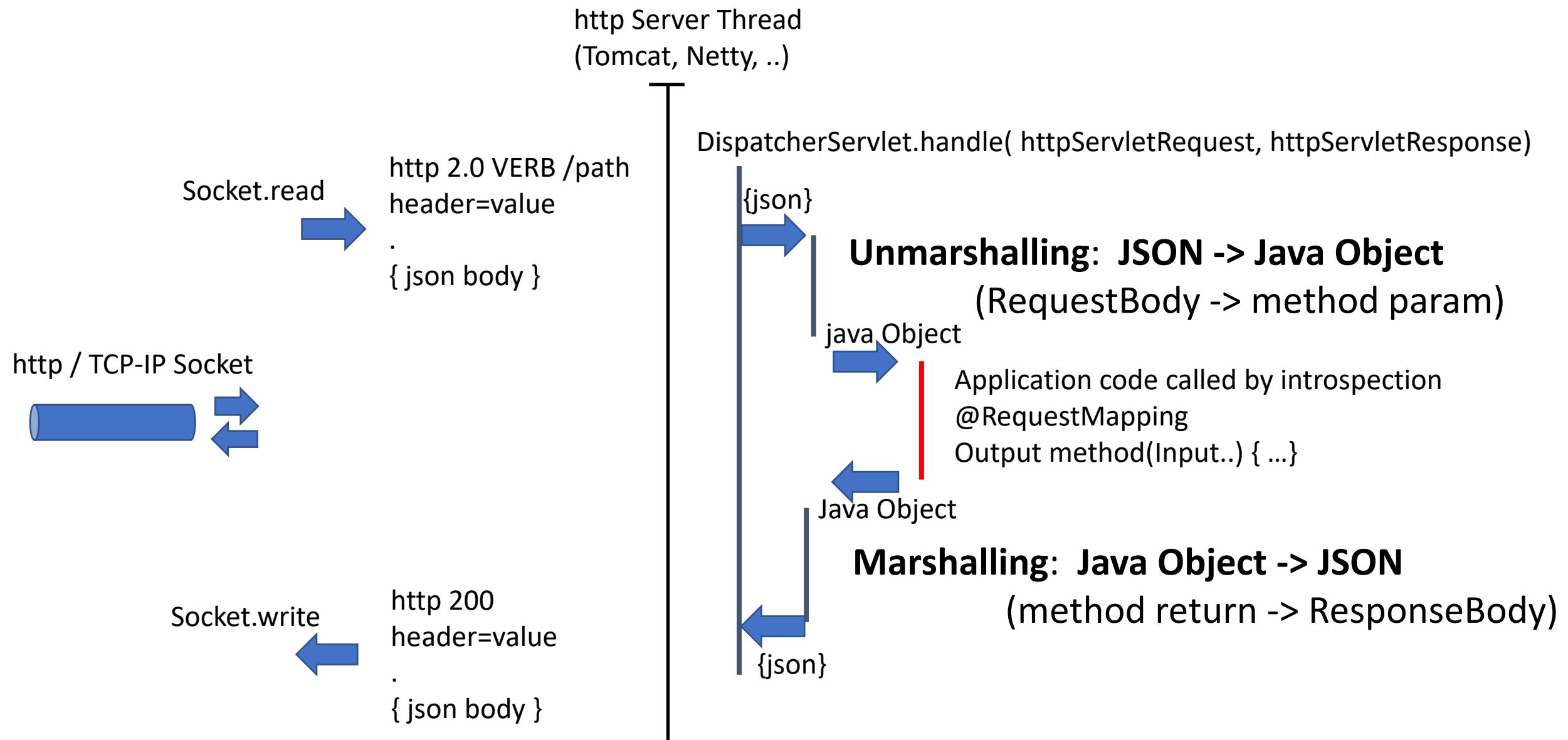
Avro Message, Avro Data-File

PARQUET File                    for ULTRA compression of millions of rows (dictionnary, incremental, filter..)

java.io.Serializable            Contains serialVersionUID + fully-qualified Names + field name / types  
                                  Use « Kryo » instead of « java.io. » for typed / schema-less messages !  
                                  (better performances )



# Http Json Request <-> Java Object method



# JavaScript <-> Json (JavaScript Object Notation) <-> Java

## Script (untyped interpreter)

```
let object = {  
    name: 'arnaud',  
    skills: [ 'IT', 'math' ]  
};  
console.log(`Hello ${object.name}`);  
  
// JS untyped: Any: map<String,Any>  
let anotherObj: any = new Object();  
// anotherObj.prototype ... JS dark-magic  
anotherObj.name = 'fabien';  
anotherObj['skills'] = [ 'IT', 'other' ];
```

## DATA format (no schema)

```
{  
    "name": "arnaud",  
    "skills": [ "IT", "math" ]  
}
```

## Langage (typed)

```
public class User {  
    public String name;  
    public List<String> skills;  
}
```



{JSON}

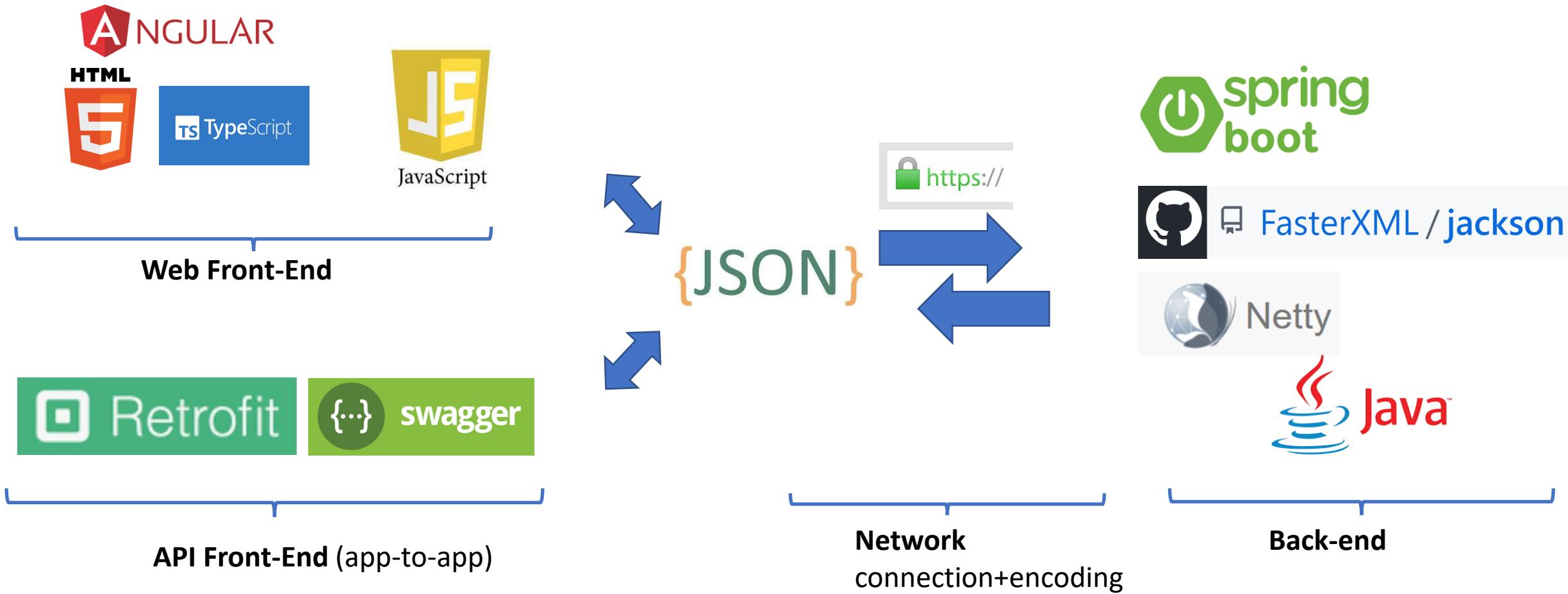


Web Front-End

Network  
encoding

Back-end

# http JSON : Open & DeFacto Standard for Portability, Simplicity, Frameworks



# Http Request <-> Spring Java Mappings

## @RequestMapping, @Get|Post|..}Mapping, @RequestBody ..

```
@PostMapping  
public TodoDTO postTodo(  
    @RequestBody TodoDTO req // => from outside, spring dispatcher...  
                           // request body as json text, is converted to java Object using Jackson  
) {  
    Log.info("http POST /api/todo");  
    TodoDTO res = service.createTodo(req);  
    return res;  
}  
  
@GetMapping("/{id}")  
public TodoDTO get(@PathVariable("id") int id) {  
    TodoDTO res = service.get(id);  
    return res;  
} // => outside, spring dispatcher... return java Object is converted to json using Jackson
```

# Equivalent Explicit Json Unmarshalling



```
@PostMapping  
public TodoDTO postTodo(  
    @RequestBody TodoDTO req) {  
    Log.info("http POST /api/todo");  
    TodoDTO res = service.createTodo(req);  
    return res;  
}  
  
@Autowired  
ObjectMapper jsonMapper;  
  
// equivalent  
@PostMapping(consumes = "application/json")  
public TodoDTO postTodo2(  
    @RequestBody byte[] reqBodyContent) throws Exception {  
    Log.info("http POST /api/todo");  
    TodoDTO req = jsonMapper.readValue(reqBodyContent, TodoDTO.class);  
    TodoDTO res = service.createTodo(req);  
    return res;  
}
```

# Equivalent Explicit Json Marshalling

```
@GetMapping("/{id}")
public TodoDTO get(@PathVariable("id") int id) {
    return service.get(id);
}

@Autowired
ObjectMapper jsonMapper;

// implicit equivalent..
@GetMapping(path = "/equivalent1/{id}",
    produces = "application/json")
public byte[] get1(@PathVariable("id") int id) throws JsonProcessingException {
    TodoDTO res = service.get(id);
    return jsonMapper.writeValueAsBytes(res);
}
```

# Explicit Equivalent, with http Status + Headers

```
// implicit equivalent.. with extra header
@GetMapping(path = "/equivalent2/{id}",
    produces = "application/json")
public ResponseEntity<byte[]> get2(@PathVariable("id") int id) throws JsonProcessingException {
    TodoDTO res = service.get(id);
    byte[] content = jsonMapper.writeValueAsBytes(res);
    return ResponseEntity.status(HttpStatus.OK)
        .header("some-response-header", "value")
        .body(content);
}

// implicit equivalent.. with extra header
@GetMapping(path = "/equivalent2/{id}",
    produces = "application/json")
public void get2(@PathVariable("id") int id,
    HttpServletResponse serlvetResponse
) throws IOException {
    TodoDTO res = service.get(id);
    byte[] content = jsonMapper.writeValueAsBytes(res);
    serlvetResponse.setStatus(200);
    serlvetResponse.addHeader("some-response-header", "value");
    serlvetResponse.getOutputStream().write(content);
}
```

# Naive (Not working) @Entity to JSON

```
@RestController  
@RequestMapping("/api/not-working/todo")  
@Transactional  
public class NaiveStupidBuggedRestController {  
  
    @Autowired  
    private TodoRepository repository;  
  
    @GetMapping("/{id}")  
    public TodoEntity get(@PathVariable("id") int id) {  
        TodoEntity entity = repository.getById(id);  
        return entity;  
    }  
}
```

Entity object invalid  
After transaction commit  
(managed lifecycle)

Entity class maybe  
not JSON compatible

# Why Not using 1 class Entity = DTO ?

0/ does NOT work except on simplicitic entities (Transaction)

1/ internal database is PRIVATE, implementation specific  
!= external API is publicly specified

2/ Decoupling helps evolutivity

3/ projection-cuts for complex Entities graph  
with cyclic dependencies parent-child (@ManyToOne(mappedBy..) )

4/ several DTO classes/APIs for a single Entity class

5/ Do not use HACKS like @JsonIgnore... (see 0/, 1/, 2/, 3/, 4/ )

# Example Entity class .. Do not export all

```
@Entity  
@Getter @Setter  
public class UserEntity {  
  
    @Id  
    private String email;  
  
    private String firstName;  
    private String lastName;  
  
    @Column(unique = true)  
    private String pseudo;  
  
    private String password;  
  
    private byte[] photo;  
}
```

Personnal data  
Do not publish

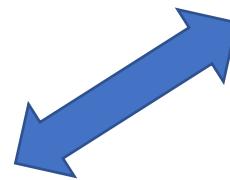
Critical Security data  
never publish !!  
( Except for owner )

High volume data  
export only explicitly demanded

# 1 Entity class <-> Several specific DTO classes

```
@Entity  
@Getter @Setter  
public class UserEntity {  
  
    @Id  
    private String email;  
  
    private String firstName;  
    private String lastName;  
  
    @Column(unique = true)  
    private String pseudo;  
  
    private String password;  
  
    private byte[] photo;  
}
```

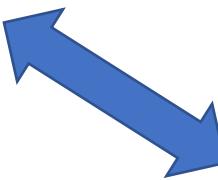
Default mapping



Owner personal view



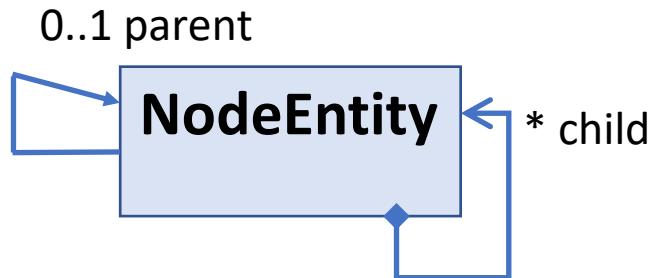
Public detailed view (high data volume)



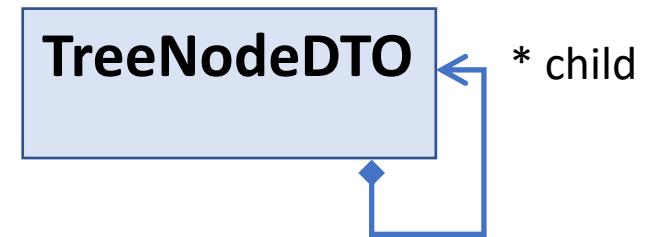
```
@Data  
public class UserLightDTO {  
  
    public String firstName;  
    public String lastName;  
    public String pseudo;  
}  
  
@Data  
public class SecuredUserDetailDTO {  
    public String email;  
    public String firstName;  
    public String lastName;  
    public String pseudo;  
  
    // computed from group, settings, ...  
    public List<String> grantedPermissions;  
}  
  
@Data  
public class SecuredUserDetailDTO {  
    public String pseudo;  
    public byte[] photo;  
}
```

# Entity -> Projection DTOs = « Cutting » relations

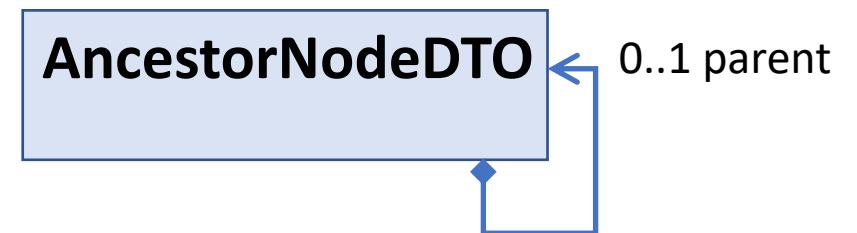
```
@Entity  
@Getter @Setter  
public class NodeEntity {  
  
    @Id  
    private String path;  
  
    // parent -> child relationship  
    @OneToMany(fetch = FetchType.LAZY, mappedBy = "parent")  
    private List<NodeEntity> child;  
  
    // child -> parent (inverse) relationship  
    @ManyToOne(fetch = FetchType.LAZY)  
    private NodeEntity parent;  
}
```



Projection Parent->Children (recursive)

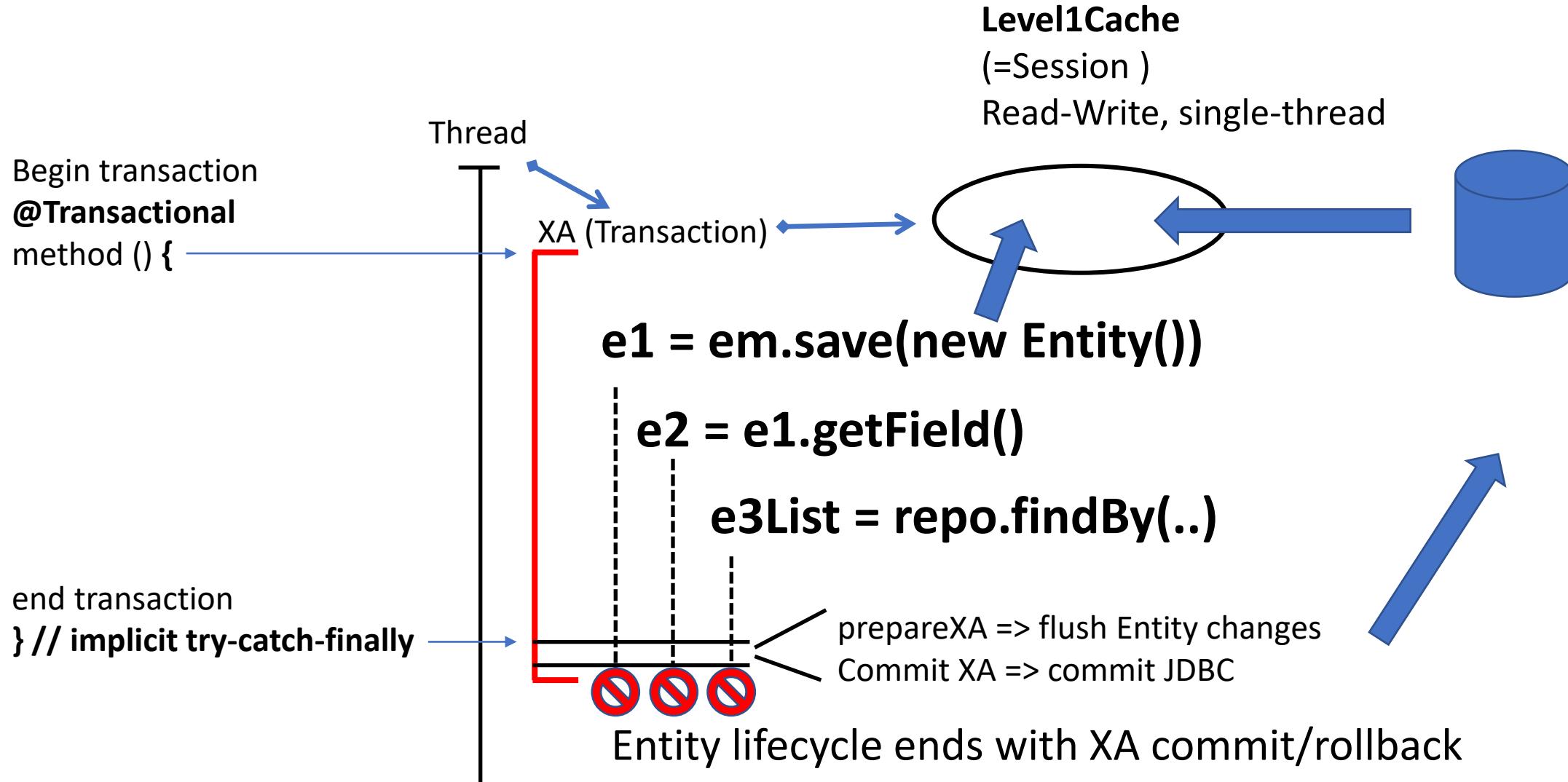


Projection Child->Parent (recursive)

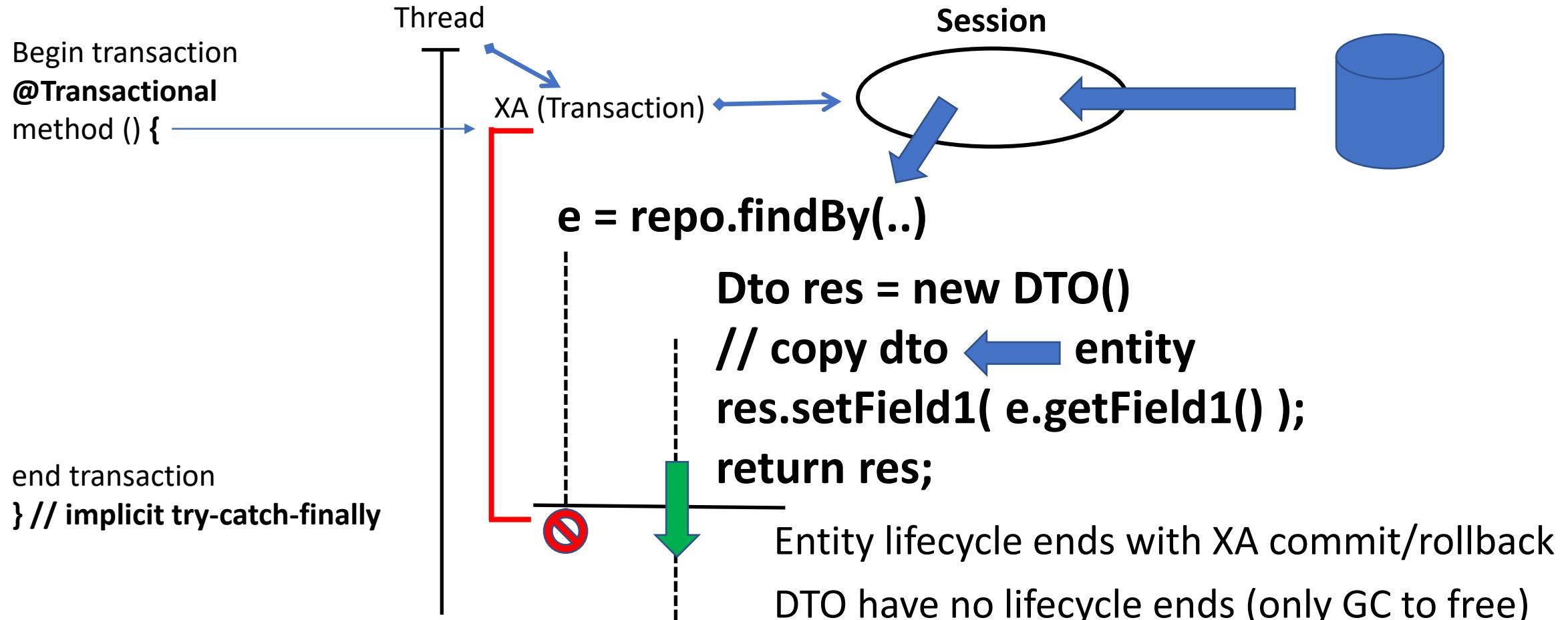


Reminder Part 2 ... Entity lifecycle in Session

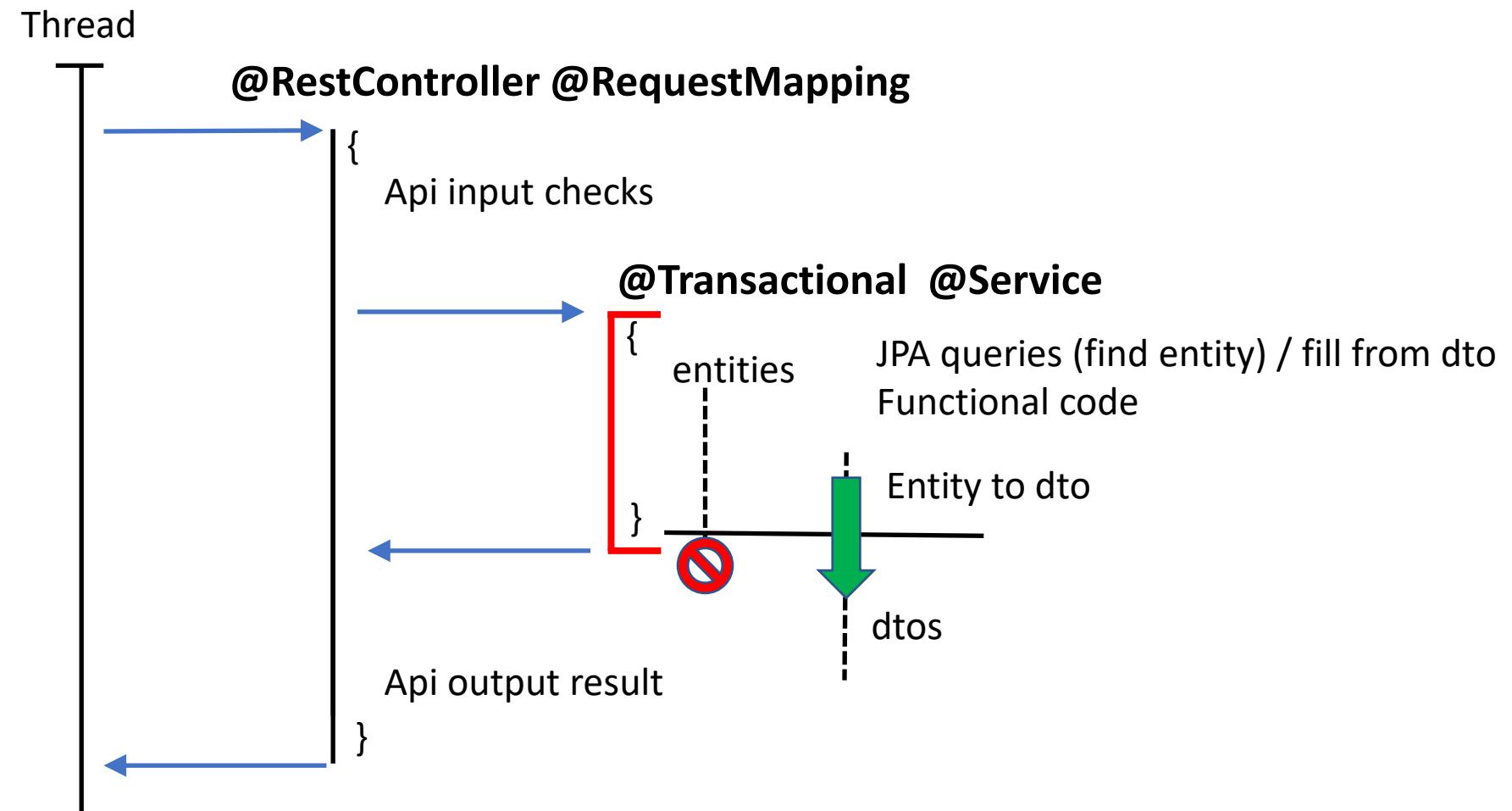
# Entity : Lifecycle managed by Session (Transaction)



# Copy Entity data to Transfer before Commit



2 Rules : a/ use DTOs != Entities  
b/ use RestController != (Transactional) Service



# But it seems to works?? (in « normal » cases)

```
INFO j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'  
WARN JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. This means, therefore, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning  
TNFO o.s.h.a.w.s.WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]
```

Therefore, database queries may be performed during view rendering. Explicitly configure `spring.jpa.open-in-view` to disable this warning



## Bad default value in springboot

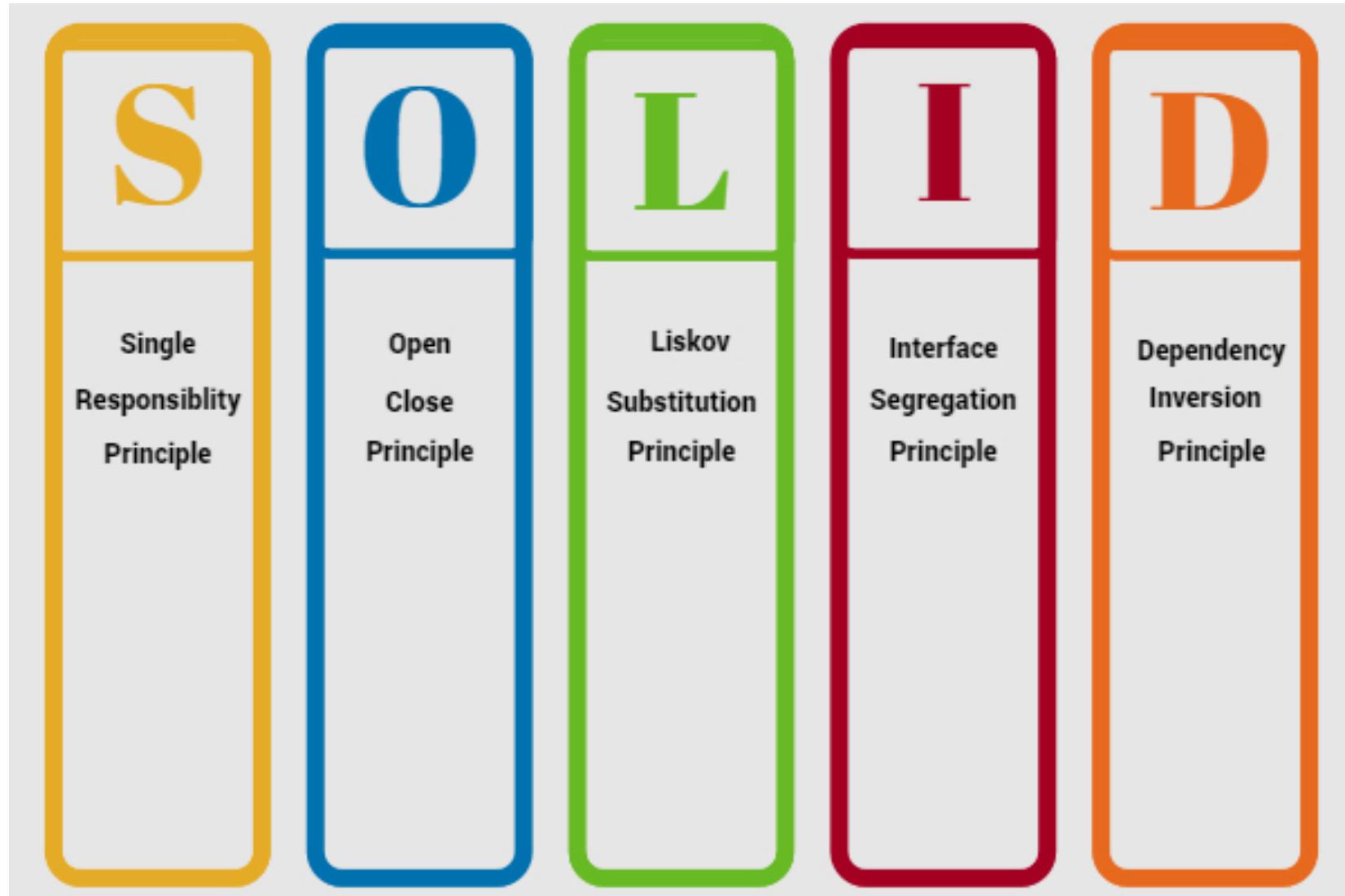
To please newbies trying JPA with left hands  
Springboot explicitly WARN to change it  
Do not simply remove the warn !!! Code is buggy

# Controller Method Pattern

## { unmarshal / delegate / marshall }

```
@PostMapping  
public ResponseDTO postTodo(  
    @RequestBody TodoDTO req) {  
    // step 1/3: unmarshal, check inputs, convert, logs...  
    long start = System.currentTimeMillis();  
    Log.info("http POST /api/todo");  
  
    // step 2/3: delegate to service  
    TodoDTO res = service.createTodo(req);  
  
    // step 3/3: convert, format output, logs...  
    Log.info(.. done http POST, took " + (System.currentTimeMillis()-start) + " ms");  
    return new ResponseDTO(res.id, res.label);  
}
```

# « solid » principles



SOLID RestController .... S = Single

A RestController does only 1 thing :

**controls (maps) http Rest requests to Java methods**

... delegate all others things to injected Service

# Typical CRUD Rest Controller

```
@RestController
@RequestMapping("/api/todo")
@Slf4j
public class TodoRestController {
    @Autowired
    private TodoService service;

    @GetMapping()
    public List<TodoDTO> list() {
        List<TodoDTO> res = service.list();
        return res;
    }

    @GetMapping("/{id}")
    public TodoDTO get(@PathVariable("id") int id) {
        TodoDTO res = service.get(id);
        return res;
    }

    @PostMapping
    public TodoDTO postTodo(@RequestBody TodoDTO req) {
        log.info("http POST /api/todo");
        TodoDTO res = service.createTodo(req);
        return res;
    }

    @PutMapping
    public TodoDTO putTodo(@RequestBody TodoDTO req) {
        log.info("http PUT /api/todo");
        TodoDTO res = service.updateTodo(req);
        return res;
    }

    @DeleteMapping("/{id}")
    public TodoDTO deleteTodo(@PathVariable("id") int id) {
        log.info("http DELETE /api/todo");
        TodoDTO res = service.deleteTodo(id);
        return res;
    }
}
```

# Typical CRUD Transactional Service (1/2)

```
@Service  
@Transactional  
public class TodoService {  
  
    @Autowired  
    private TodoRepository repository;  
  
    @Autowired  
    private DtoConverter dtoConverter;  
  
    public List<TodoDTO> list() {  
        List<TodoEntity> entities =  
            repository.findAll();  
        return entity2Dtos(entities);  
    }  
  
    public TodoDTO get(int id) {  
        TodoEntity entity = repository.getById(id);  
        return entity2Dto(entity);  
    }
```

```
    public TodoDTO createTodo(TodoDTO req) {  
        TodoEntity res = repository.save(dto2Entity(req));  
        return entity2Dto(res);  
    }  
  
    public TodoDTO updateTodo(TodoDTO req) {  
        TodoEntity entity = repository.getById(req.id);  
        entity.setLabel(req.label);  
        entity.setPriority(req.priority);  
        return entity2Dto(entity);  
    }  
  
    public TodoDTO deleteTodo(int id) {  
        TodoEntity entity = repository.getById(id);  
        repository.delete(entity);  
        return entity2Dto(entity);  
    }
```

# Typical CRUD Service (2/2)

## entity2Dto / dto2Entity

```
public TodoDTO entity2Dto(TodoEntity src) {
    TodoDTO res = new TodoDTO();
    res.id = src.getId();
    res.label = src.getLabel();
    res.priority = src.getPriority();
    // other fields...
    return res;
}

public TodoEntity dto2Entity(TodoDTO src) {
    TodoEntity res = new TodoEntity();
    res.label = src.label;
    res.priority = src.priority;
    // other fields...
    return res;
}

public List<TodoDTO> entity2Dtos(Collection<TodoEntity> src) {
    return src.stream().map(e -> entity2Dto(e)).collect(Collectors.toList());
}
```

# entity2Dto / dto2Entity using generic signature and « .class »

```
protected TodoDTO entity2Dto(TodoEntity src) {  
    return dtoConverter.map(src, TodoDTO.class);  
}  
  
protected List<TodoDTO> entity2Dtos(Collection<TodoEntity> src) {  
    return dtoConverter.mapAsList(src, TodoDTO.class);  
}  
  
protected TodoEntity dto2Entity(TodoDTO src) {  
    return dtoConverter.map(src, TodoEntity.class);  
}
```

# Using Orika MapperFacade

```
@Component
public class DtoConverter {

    private MapperFacade mapper = createMapper();

    private MapperFacade createMapper() {
        MapperFactory mapperFactory = new DefaultMapperFactory.Builder().build();
        return mapperFactory.getMapperFacade();
    }

    public <S, D> D map(S sourceObject, Class<D> destinationClass) {
        return mapper.map(sourceObject, destinationClass);
    }

    public <S, D> List<D> mapAsList(Iterable<S> source, Class<D> destinationClass) {
        return mapper.mapAsList(source, destinationClass);
    }

}
```

# Orika

```
<dependency>
  <groupId>ma.glasnost.orika</groupId>
  <artifactId>orika-core</artifactId>
  <version>${orika.version}</version>
</dependency>
```

<https://search.maven.org/search?q=q=orika-core>



Maven Central Repository Search    Quick Stats

orika-core

Group ID

Artifact ID

Latest Version

[ma.glasnost.orika](#)

orika-core

1.5.4

(29)

# <https://github.com/orika-mapper/orika>

The screenshot shows the GitHub repository page for `orika-mapper/orika`. The page has a dark theme. At the top, there is a navigation bar with icons for search, pull requests, issues, marketplace, and explore. Below the navigation bar, the repository name `orika-mapper / orika` is displayed, along with its status as `Public`, the number of `Watch`ers (73), `Fork`s (255), and `Star`s (1.1k). It is noted that the repository is `forked from elaatifi/orika`. The main navigation tabs include `Code` (selected), `Issues` (141), `Pull requests` (5), `Actions`, `Projects`, `Wiki`, and `...`. Below the tabs, there is a dropdown for the `master` branch, a `Go to file` button, an `Add file` button, and a `Code` button. A message box indicates that the `master` branch is 592 commits ahead and 1 commit behind the `elaatifi:master` branch. To the right, there is an `About` section with a description of the project as a "Simpler, better and faster Java bean mapping framework" and a link to the documentation at `orika-mapper.github.io/orika-docs/`. The `Code` section shows a list of recent commits:

| Author | Commit Message                                       | Date                         | Commits          |
|--------|--|------------------------------|------------------|
|        | <code>tomashanley-toast and elaatifi Upd...</code>   | <code>on Oct 19, 2021</code> | <code>995</code> |
|        | <code>bump project version to 1.6.0</code>           | <code>4 months ago</code>    |                  |
|        | <code>bump project version to 1.6.0</code>           | <code>4 months ago</code>    |                  |
|        | <code>extract Janino into a separate mod...</code>   | <code>4 months ago</code>    |                  |
|        | <code>ignores some tests to fix Java 17 bu...</code> | <code>4 months ago</code>    |                  |

On the right side, there are buttons for `java` and `mapper`, and links to `Readme`, `Apache-2.0 License`, `1.1k stars`, `73 watching`, and `255 forks`.

<http://orika-mapper.github.io/orika-docs/>

The screenshot shows the 'User Guide' page of the Orika documentation. At the top, there's a header with the text 'Orika *simpler, lighter and faster Java bean mapping*' and a large 'User Guide' title. Below the title is a yellow box containing an 'Important!' note about the guide being a work in progress and instructions for feedback. The main content area includes sections for 'About', 'Approach', and 'Topics'. The 'Topics' section lists various configuration and usage topics with their descriptions.

#### Important!

This guide is (like Orika) a work in progress; any feedback in the way of suggestions or questions or problems can be posted to our project issues site, currently hosted at [Google Code](#)

## About

As developers we have to provide solutions to business problems, and we want to use the time in our disposition to do what really matter. In our days, enterprise applications became more and more complex, with a lot of architecture and design constrains. Design constraints that will produce a considerable amount of mechanical work. A lot of Open Source projects put in our hands some great tools to face such complexity, such *Spring*, *Guice*, *Hibernate*, *Wicket*, ... and we have a lot of options available to solve each particular part of the overall problem. However, with all of these different tools/frameworks/libraries, it is common to find that we need to convert objects into different formats to accommodate different APIs, or we may even need to convert formats between different architectural layers of our own for design reasons; to accomplish this, we are left to the rather boring task of writing mapping code to copy the values from one type to another.

In a medium to large project, such mapping code can reach a considerable effort of mechanical (boring) work which can be difficult to maintain, test, and debug.

## Approach

Orika attempts to perform this tedious work for you, with little measurable tradeoff on performance

It will automatically collect meta-data of your classes to generate mapping objects which can be used together to copy data from one object graph to another, recursively. Orika attempts to provide many convenient features while remaining relatively simple and open -- giving you the possibility to extend and adapt it to fit your needs.

## Topics

### Introduction

getting started with Orika

### Declarative Mapping Configuration

using the fluent-style ClassMapBuilder API

### Advanced Mapping Configurations

beyond the basics of ClassMapBuilder API

### MapperFactory Configuration

settings to customize default mapping behavior

### Custom Converters

explicitly defining conversion from one type to another

### Object Factories

customize object instantiation

### Custom Mappers

explicitly define how properties are copied from one object to another

### Filters

customize dynamic mapping behavior

### Converters, Mappers, and ObjectFactories

when to choose one over the others

### Extending Orika

advanced customizations

### Performance Tuning

recommendations for optimal performance

### Troubleshooting

what to do when something goes wrong

### FAQ

frequently asked questions

### Examples

cooking with Orika

# Orika Intro ... customizing Class Mapper

```
1. mapperFactory.classMap(PersonSource.class, PersonDestination.class)
2.     .field("firstName", "givenName")
3.     .field("lastName", "sirName")
4.     .byDefault()
5.     .register();
```

Convention over configuration

Override custom  
Bi-Directional field Mapping  
= fieldAtoB + fieldBToA

```
.fieldAtoB("name", "fullName")
```

Explicit Single-Directional  
fieldAtoB != fieldBToA

```
.exclude("name")
```

```
.constructorA("name", "id")
```

# Customizing... mapping different structures

```
.field("name.first", "firstName")
```

Nested field class <-> flat field

```
.field("nameParts[0]", "firstName")
.field("nameParts[1]", "lastName")
```

List<..> by index <-> flat field

```
.field("nameParts['first']", "firstName")
.field("nameParts[\\"last\\"]", "lastName")
```

Map<Key,..> <-> flat field

```
.field("names{fullName}", "personalNames{key}")
.field("names{}", "personalNames{value}")
```

List<Nested field class> <-> Map<String,..>

# Customizing Properties Naming Conventions

Naming convention  
for all Properties

```
.propertyResolverStrategy(new ElementPropertyResolver())  
  
public static class ElementPropertyResolver extends IntrospectorPropertyResolver {  
    protected Property getProperty(java.lang.reflect.Type type, String expr,  
        boolean isNestedLookup, Property owner) throws MappingException {
```

Naming convention  
for 1 Property

*In-line property syntax:*

```
«name» :{ «getter» | «setter» [ | type= «type» ] }
```

Details

# Customizer Mapper

override default Class to Class  
Custom Mapper  
(handle specific fields only)

```
mapperFactory.classMap(Source.class, Destination.class)
    .byDefault()
    .customize(
        new CustomMapper<Source, Destination> {
            public void mapAtoB(A a, B b, MappingContext context) {
                // add your custom mapping code here
            }
        }
    )
    .register();
```

Class to Class Full  
Custom Mapper  
(handle all fields)

```
factory.registerMapper(new MyCustomMapper());
```

Details

# Custom Per Type Converter (apply to all type1<->type2 properties)

ConverterFactory converterFactory = mapperFactory.getConverterFactory();  
converterFactory.registerConverter(new MyConverter());

```
public class MyConverter extends BidirectionalConverter<Date,MyDate> {  
  
    public MyDate convertTo(Date source, Type<MyDate> destinationType) {  
        // convert in one direction  
    }  
  
    public Date convertFrom(MyDate source, Type<Date> destinationType) {  
        // convert in the other direction  
    }  
}
```

Details

# Custom « new » Object Factory

```
mapperFactory.registerObjectFactory(new PersonFactory(), Person.class);
```

```
public class PersonFactory implements ObjectFactory<Person> {

    public Person create(Object source, Type<Person> destinationType) {
        Person person = new Person();
        // set the default address
        person.setAddress(new Address("Morocco", "Casablanca"));
        return person;
    }
}
```

Details

## More customizations ... Filter

```
factory.registerFilter(new MyCustomFilter());
```

customize dynamic mapping behavior

Details

# Code Generator Options

Details

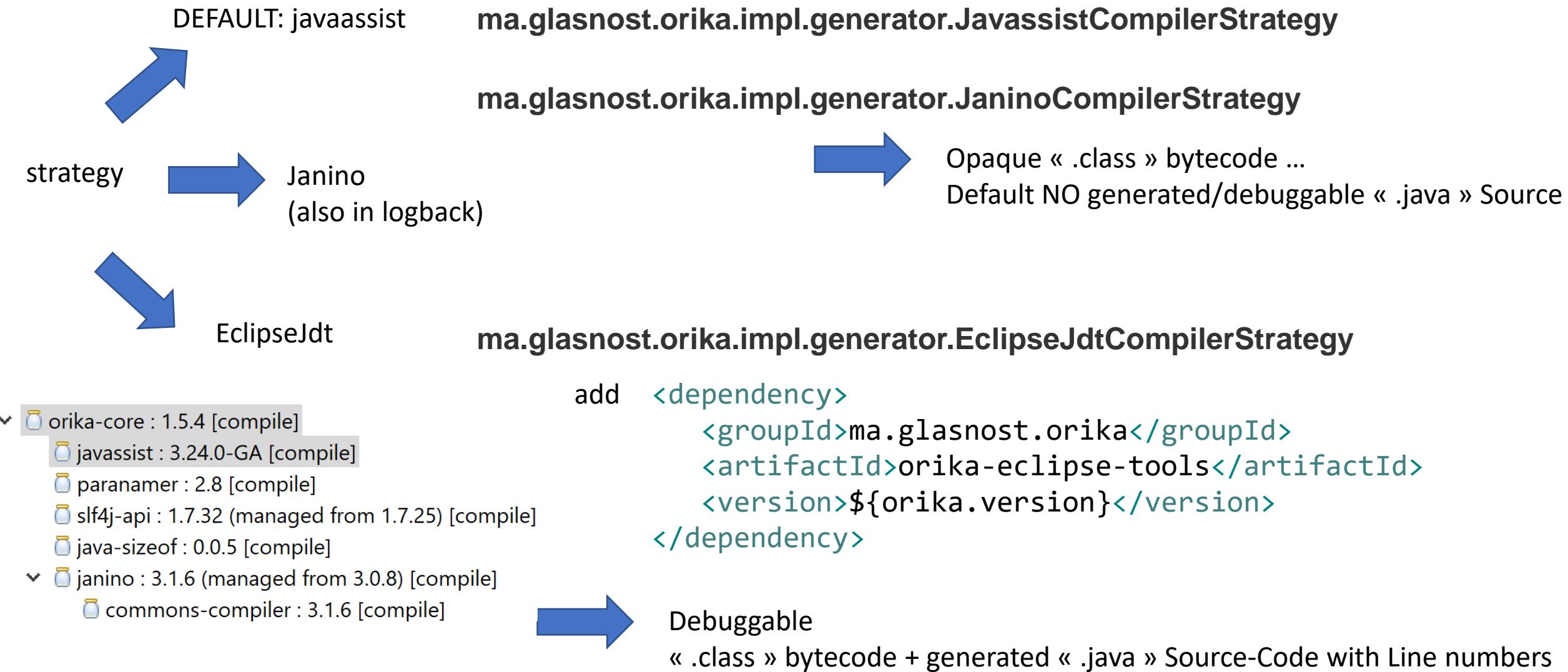
```
/*
 * Specification encapsulates the logic to generate code for mapping comparing a pair of types
 */
public interface Specification {

    /**
     * Tests whether this Specification applies to the specified FieldMap
     */
    boolean appliesTo(FieldMap fieldMap);

    /**
     * Generates code for a boolean equality test between the two variable types,
     * where are potentially unrelated.
     */
    String generateEqualityTestCode(FieldMap fieldMap, VariableRef source, VariableRef destination, SourceCodeContext code);

    /**
     * Generates code to map the provided field map
     */
    String generateMappingCode(FieldMap fieldMap, VariableRef source, VariableRef destination, SourceCodeContext code);
}
```

# Code Generated ??? YES !!!



# Javaassist + register «ClassPool in ClassLoader

<https://github.com/orika-mapper/orika/blob/master/core/src/main/java/ma/glasnost/orika/impl/generator/JavassistCompilerStrategy.java#L134>

```
74     this.classPool = new ClassPool();
75     this.classPool.appendSystemPath();
76
77     this.classPool.insertClassPath(new ClassClassPath(this.getClass()));
```

```
127     private boolean registerClassLoader(ClassLoader cl) {
128         Boolean found = referencedLoaders.get(cl);
129         if (found == null) {
130             synchronized (cl) {
131                 found = referencedLoaders.get(cl);
132                 if (found == null) {
133                     referencedLoaders.put(cl, Boolean.TRUE);
134                     classPool.insertClassPath(new LoaderClassPath(cl));
135                 }
136             }
137         }
138         return found == null || !found;
139     }
```

# Writing Generated Classes..

[https://github.com/orika-mapper/orika/blob/master/  
core/src/main/java/ma/glasnost/orika/impl/generator/CompilerStrategy.java#L53](https://github.com/orika-mapper/orika/blob/master/core/src/main/java/ma/glasnost/orika/impl/generator/CompilerStrategy.java#L53)

...

```
53     protected final boolean writeSourceFiles;  
54     protected final boolean writeClassFiles;  
55     protected final String pathToWriteSourceFiles;  
56     protected final String pathToWriteClassFiles;
```

Using System.properties

```
ma.glasnost.orika.writeSourceFiles  
ma.glasnost.orika.writeClassFiles  
ma.glasnost.orika.writeSourceFilesToPath  
ma.glasnost.orika.writeClassFilesToPath
```

# First call map() => generating Mapper for (ClassA,ClassB)

The screenshot shows an IDE interface with several tabs at the top: User.java, test-springb..., TodoDTO.java, TodoEntity.java, OrikaMapperT..., DbDatainiti..., DtoConverte..., MapperFacade..., Vari..., Bre..., and Exp... . The main code editor window displays a JUnit test class:

```
16@User.java 17 @Test 18 public void testMap() { 19     System.setProperty("ma.glasnost.orika.writeSourceFiles", "true"); 20     val mapperFactoryBuilder = new DefaultMapperFactory.Builder(); 21 22     MapperFactory mapperFactory = mapperFactoryBuilder.build(); 23     MapperFacade mapperFacade = mapperFactory.getMapperFacade(); 24 25     TodoEntity src = new TodoEntity(); 26     src.setId(1); 27     src.setLabel("learn orika"); 28 29     // first convert => generate bytecode for converter(+ writeSourceFiles=true) 30     TodoDTO resDTO = mapperFacade.map(src, TodoDTO.class); 31 32     Assertions.assertEquals(src.getId(), resDTO.id); 33 34     // second convert => use cached converter (ClassLoader class) 35     resDTO = mapperFacade.map(src, TodoDTO.class); 36 }
```

The code uses assertions to check if the source entity's ID is correctly mapped to the destination DTO's ID. Below the code editor is a console window showing the execution of the test:

```
Console × Problems × Error Log × Debug Shell × Search × Call Hierarchy  
OrikaMapperTest.testMap [JUnit] C:\apps\jdk\jdk-8\bin\javaw.exe (27 janv. 2022 à 14:46:48)  
14:47:49.374 [main] DEBUG ma.glasnost.orika.impl.DefaultMapperFactory - No mapper registered for (TodoEntity, TodoDTO): attempting to generate  
14:47:50.079 [main] DEBUG ma.glasnost.orika.metadata.ClassMapBuilder - ClassMap created:  
    ClassMapBuilder.map(TodoEntity, TodoDTO)  
        .field( id(int), id(int) )  
        .field( label(String), label(String) )  
        .field( priority(int), priority(int) )  
14:47:51.128 [main] DEBUG ma.glasnost.orika.impl.generator.JavassistCompilerStrategy - Source file written to C:\arn\  
    -classes\ma\glasnost\orika\generated\Orika TodoDTO TodoEntity Mapper1026685228480800$0.java  
14:47:53.258 [main] DEBUG ma.glasnost.orika.impl.generator.MapperGenerator - Generating new mapper for (TodoEntity, TodoDTO)  
    Orika_TodoDTO_TodoEntity_Mapper1026685228480800$0.mapAtoB(TodoEntity, TodoDTO) {  
        Field(id(int), id(int)) : copying int by reference  
        Field(label(String), label(String)) : copying String by reference  
        Field(priority(int), priority(int)) : copying int by reference  
    }  
    Orika_TodoDTO_TodoEntity_Mapper1026685228480800$0.mapBtoA(TodoDTO, TodoEntity) {  
        Field(id(int), id(int)) : copying int by reference  
        Field(label(String), label(String)) : copying String by reference  
        Field(priority(int), priority(int)) : copying int by reference  
    }  
14:48:36.943 [main] DEBUG ma.glasnost.orika.impl.MapperFacadeImpl - MappingStrategy resolved and cached:  
    Inputs:[ sourceClass: com.example.demo.rest.TodoEntity, sourceType: null, destinationType: class com.example.demo.rest.TodoDTO]  
    Resolved:[ strategy: InstantiateAndUseCustomMapperStrategy, sourceType: TodoEntity, destinationType: TodoDTO, mapper: GeneratedMapper<TodoEntity, TodoDTO> {
```

# Generated code.. (file not visible in Eclipse?)

```
package ma.glasnost.orika.generated;

public class Orika_TodoDTO_TodoEntity_Mapper1026685228480800$0 extends ma.glasnost.orika.impl.GeneratedMapperBase {

    public void mapAtoB(java.lang.Object a, java.lang.Object b, ma.glasnost.orika.MappingContext mappingContext) {

        super.mapAtoB(a, b, mappingContext);

        // sourceType: TodoEntity
        com.example.demo.rest.TodoEntity source = ((com.example.demo.rest.TodoEntity)a);
        // destinationType: TodoDTO
        com.example.demo.rest.TodoDTO destination = ((com.example.demo.rest.TodoDTO)b);

        destination.id = ((int)source.getId());
        destination.label = ((java.lang.String)source.getLabel());
        destination.priority = ((int)source.getPriority());
        if(customMapper != null) {
            customMapper.mapAtoB(source, destination, mappingContext);
        }
    }

    public void mapBtoA(java.lang.Object a, java.lang.Object b, ma.glasnost.orika.MappingContext mappingContext) {

        super.mapBtoA(a, b, mappingContext);

        // sourceType: TodoDTO
        com.example.demo.rest.TodoDTO source = ((com.example.demo.rest.TodoDTO)a);
        // destinationType: TodoEntity
        com.example.demo.rest.TodoEntity destination = ((com.example.demo.rest.TodoEntity)b);

        destination.setId(((int)source.id));
        destination.setLabel(((java.lang.String)source.label));
        destination.setPriority(((int)source.priority));
        if(customMapper != null) {
```

Optimal compiled code  
Entity -> DTO  
(with extra cast)

Optimal compiled code  
DTO -> Entity  
(with extra cast)

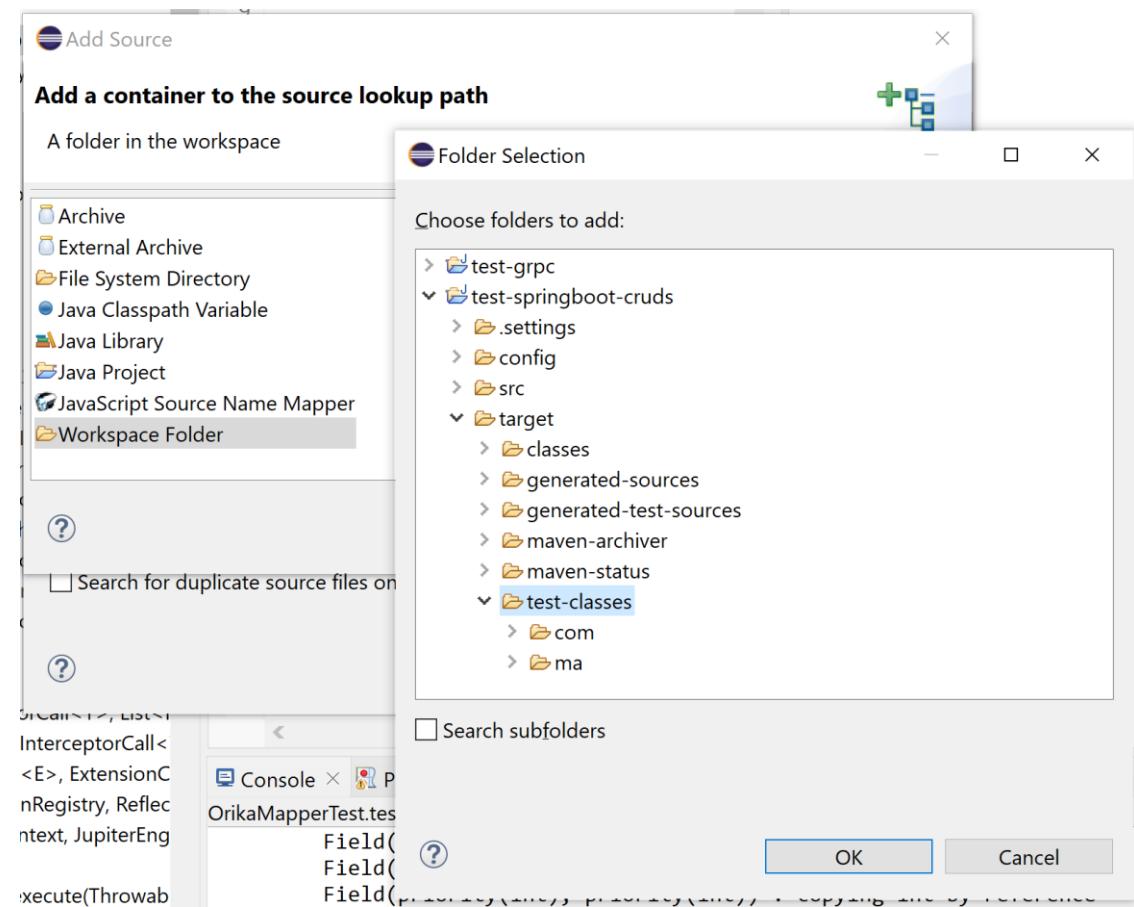
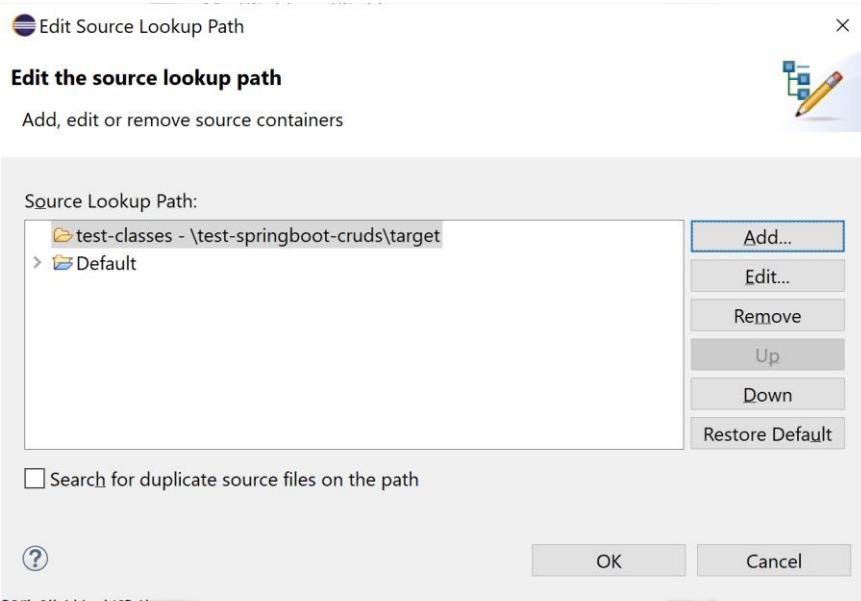
# Generated Code .. Source not visible in Eclipse

The screenshot shows the Eclipse IDE interface with the following details:

- Left Panel (Project Explorer):** Shows the current project structure and files. A test run named "OrikaMapperTest.testMap [JUnit]" is selected, which contains a "RemoteTestRunner" at localhost:54771.
- Middle Panel (TodoEntity.java):** Displays the Java code for the `TodoEntity` class. The code includes annotations like `@Entity`, `@Id`, and `@GeneratedValue`. A specific line of code, `return this.label;`, is highlighted in green.
- Right Panel (Status Bar):** Shows a message: "Source not found." Below it is a button labeled "Edit Source Lookup Path...".

```
TodoEntity.java ×
1 package com.example.demo.rest;
2
3+import javax.persistence.Entity;■
9
10 @Entity
11 @Getter @Setter
12 public class TodoEntity {
13@  @Id @GeneratedValue
14  private int id;
15
16  String label;
17  private String creationDate;
18  int priority;
19
20@ 21  public String getLabel() {
22      return this.label;
23  }
24 }
```

# Adding Source



# Debugging « Line Number » info not available! ( mode javaassist ... switch to mode EclipseJdt )

The screenshot shows the Eclipse IDE interface during a debug session. The left pane displays the stack trace, and the right pane shows the source code for `TodoEntity.java`.

Stack Trace (Left):

- `TodoEntity.getLabel() line: 21`
- `Orika_TodoDTO_TodoEntity_Mapper1027876491952400$0.mapAtoB(Object, Object, MappingContext) line: not available`
- `InstantiateAndUseCustomMapperStrategy(UseCustomMapperStrategy).map(Object, Object, MappingContext) line: 77`
- `MapperFacadeImpl.map(S, Class<D>, MappingContext) line: 672`

Source Code (Right):

```
TodoEntity.java  Orika_TodoDTO_TodoEntity_Mapper1027876491952400$0.java
1 package ma.glasnost.orika.generated;
2
3 public class Orika_TodoDTO_TodoEntity_Mapper1027876491952400$0 extends ma.glasnost.orika.MapperFacade {
4
5     public void mapAtoB(java.lang.Object a, java.lang.Object b, ma.glasnost.orika.MapperFacadeImpl sourceMapping) {
6
7         super.mapAtoB(a, b, sourceMapping);
8
9     }
10
11    // sourceType: TodoEntity
12    com.example.demo.rest.TodoEntity source = ((com.example.demo.rest.TodoEntity)a);
13    // destinationType: TodoDTO
14    com.example.demo.rest.TodoDTO destination = ((com.example.demo.rest.TodoDTO)b);
15
16    destination.id = ((int)source.getId());
17    destination.label = ((java.lang.String)source.getLabel());
18    destination.priority = ((int)source.getPriority());
19    if(customMapper != null) {
20        customMapper.mapAtoB(source, destination, sourceMapping);
21    }
22}
23
24
25    public void mapBtoA(java.lang.Object a, java.lang.Object b, ma.glasnost.orika.MapperFacadeImpl sourceMapping) {
26
27        super.mapBtoA(a, b, sourceMapping);
28
29    }
30}
```

A red circle highlights the line number 'not available' for the `mapAtoB` call. A blue arrow points from a speech bubble containing the text 'Supposed to be here <.getLabel()> Line 18 ... ??' to this line number.

# Activating EclipseJdtCompilerStrategy

```
@Test
public void testMap2() {
    System.setProperty("ma.glasnost.orika.writeSourceFiles", "true");
    val mapperFactoryBuilder = new DefaultMapperFactory.Builder();

    // java -Dma.glasnost.orika.compilerStrategy=ma.glasnost.orika.impl.generator.EclipseJdtCompilerStrategy.Ecli
    // or equivalent..
    System.setProperty("ma.glasnost.orika.compilerStrategy",
        "ma.glasnost.orika.impl.generator.EclipseJdtCompilerStrategy.EclipseJdtCompilerStrategy");
    // or equivalent..
    mapperFactoryBuilder.compilerStrategy(new EclipseJdtCompilerStrategy());

    MapperFactory mapperFactory = mapperFactoryBuilder.build();
    MapperFacade mapperFacade = mapperFactory.getMapperFacade();
```

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** Debug, Project Explorer, Type Hierarchy, JUnit.
- Left Side:** Thread [main] (Suspended (breakpoint at line 21 in TodoEntity))
  - TodoEntity.getLabel() line: 21
  - Orika\_TodoDTO\_TodoEntity\_Mapper1028954979679100\$0.mapAtoB(Object, Object, MappingContext) line: 18 (highlighted with a green circle)
  - InstantiateAndUseCustomMapperStrategy(UseCustomMapperStrategy).map(Object, Object, MappingContext) line: 672
  - MapperFacadeImpl.map(S, Class<D>, MappingContext) line: 672
  - MapperFacadeImpl.map(S, Class<D>) line: 651
  - OrikaMapperTest.testMap2() line: 64
  - NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native method]
  - NativeMethodAccessorImpl.invoke(Object, Object[]) line: 64
  - DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 43
  - Method.invoke(Object, Object...) line: 564
  - ReflectionUtils.invokeMethod(Method, Object, Object...) line: 725
  - MethodInvocation<T>.proceed() line: 60
- Right Side:** Source code editor showing the EclipseJdtCompilerStrategy class.

```
10
11 // sourceType: TodoEntity
12 com.example.demo.rest.TodoEntity source = ((com.example.demo.rest.TodoEntity)a);
13 // destinationType: TodoDTO
14 com.example.demo.rest.TodoDTO destination = ((com.example.demo.rest.TodoDTO)b);
15
16
17 destination.id = ((int)source.getId());
18 destination.label = ((java.lang.String)source.getLabel());
19 destination.priority = ((int)source.getPriority());
20     if(customMapper != null) {
21         customMapper.mapAtoB(source, destination, mappingContext);
22     }
23 }
```

A green circle highlights the line 18 in the code, which corresponds to the highlighted line in the call stack.

# Orika = the « Simpler, faster and better Java bean mapping framework»



1/ Ultra FAST (as fast as optimized hand-written code)



2/ Ultra concise API ... convention of configuration



3/ fully customizable if needed



4/ No need to configure compile-time code-generated / IDE plugins



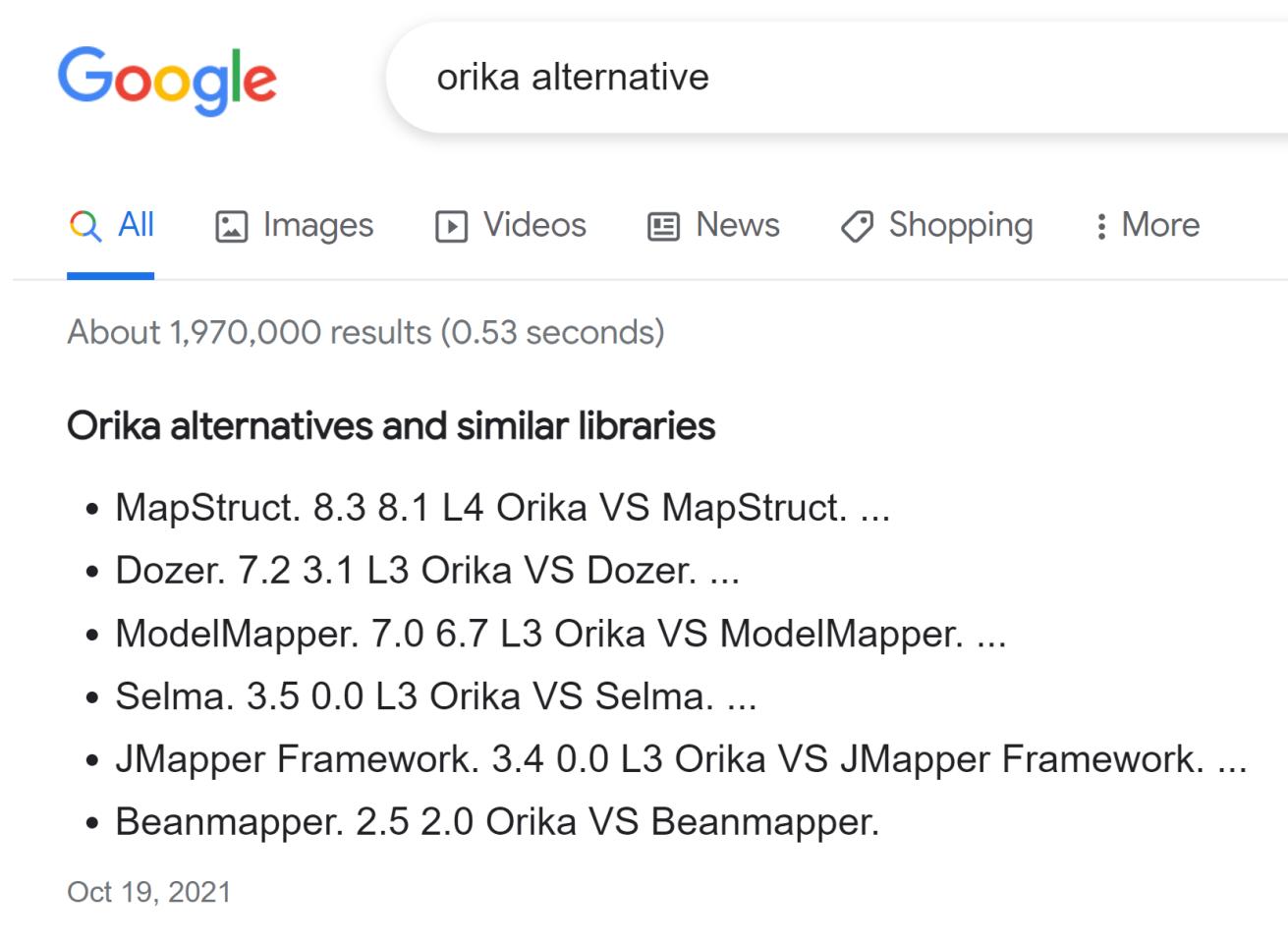
5/ can still see generated code in IDE, on demand



-- NO built-in recognized support for `findById() <-> entity` in JPA + springboot

NOT suitable for JPA « merge » / need business logic code

# Orika Alternatives



A screenshot of a Google search results page. The search query "orika alternative" is entered in the search bar. The results are filtered under the "All" category. The search took 0.53 seconds and found approximately 1,970,000 results. The first result is titled "Orika alternatives and similar libraries" and lists several alternatives:

- MapStruct. 8.3 8.1 L4 Orika VS MapStruct. ...
- Dozer. 7.2 3.1 L3 Orika VS Dozer. ...
- ModelMapper. 7.0 6.7 L3 Orika VS ModelMapper. ...
- Selma. 3.5 0.0 L3 Orika VS Selma. ...
- JMapper Framework. 3.4 0.0 L3 Orika VS JMapper Framework. ...
- Beanmapper. 2.5 2.0 Orika VS Beanmapper.

At the bottom left, the date "Oct 19, 2021" is visible.

« **MapStruct** » chosen in JHipster

- ++ code generated at compile-time
- ... need IDE plugins / config
- need to declare all methods signatures in interfaces

## Non viable legacy ideas

« **Dozer** » !!

Old, buggy, NO code generated  
Very bad performances  
Fully by introspection

# MapStruct

## (alternative chosen in Jhipster )

MapStruct

News

Documentation

Community

Development

FAQ

Code & Issues

<https://mapstruct.org/>



Java bean mappings, the easy way!

[Get started »](#)

[Download »](#)

### What is it?

MapStruct is a code generator that greatly simplifies the implementation of mappings between Java bean types based on a convention over configuration approach.

The generated mapping code uses plain method invocations and thus is fast, type-safe and easy to understand.

### Why?

Multi-layered applications often require to map between different object models (e.g. entities and DTOs). Writing such mapping code is a tedious and error-prone task. MapStruct aims at simplifying this work by automating it as much as possible.

In contrast to other mapping frameworks MapStruct generates bean mappings at compile-time which ensures a high performance, allows for fast developer feedback and thorough error checking.

### How?

MapStruct is an annotation processor which is plugged into the Java compiler and can be used in command-line builds (Maven, Gradle etc.) as well as from within your preferred IDE.

MapStruct uses sensible defaults but steps out of your way when it comes to configuring or implementing special behavior.

# <https://github.com/mapstruct/mapstruct>

The screenshot shows the GitHub repository page for `mapstruct/mapstruct`. The page has a dark theme. At the top, there is a navigation bar with links for `Pulls`, `Issues`, `Marketplace`, and `Explore`. On the right side of the navigation bar are icons for issues, pull requests, forks, stars, and a dropdown menu. Below the navigation bar, the repository name `mapstruct / mapstruct` is displayed, along with a `Public` badge. To the right of the repository name are buttons for `Watch` (134), `Fork` (674), and `Star` (4.9k). Below these buttons is a horizontal navigation bar with links for `Code`, `Issues` (321), `Pull requests` (19), `Discussions`, `Actions`, `Projects`, `Wiki`, and a `...` button. The `Code` button is highlighted with a red underline. On the left side of the main content area, there is a dropdown menu for the `master` branch, followed by buttons for `Go to file`, `Add file`, and a green `Code` button with a dropdown arrow. The main content area displays a list of recent commits. The first commit is by `JKLedzion` with the issue `#2674`, titled "Add check if method without im...", made 3 days ago with 1,517 approvals. Below this are four more commits: `.github/workflows` (issue `#2591`), `.mvn/wrapper`, `build-config`, and `core-jdk8`, all made between 4 months and 2 months ago. To the right of the commit list is a section titled `About` with the text "An annotation processor for generating type-safe bean mappers". Below this is a link to `mapstruct.org/` and a series of blue circular tags representing keywords: `java`, `mapping`, `annotation-processor`, `mapstruct`, `javabeans`, `no-reflection`, and `bean-mapping`.

mapstruct / mapstruct Public

Watch 134 Fork 674 Star 4.9k

Code Issues 321 Pull requests 19 Discussions Actions Projects Wiki ...

master Go to file Add file Code

JKLedzion #2674: Add check if method without im... 3 days ago 1,517

.github/workflows #2591 Update dependencies so tests run ... 4 months ago

.mvn(wrapper) Update maven wrapper version to 3.8.2 (...) 5 months ago

build-config [maven-release-plugin] prepare for next ... 2 months ago

core-jdk8 [maven-release-plugin] prepare for next ... 2 months ago

About

An annotation processor for generating type-safe bean mappers

mapstruct.org/

java mapping

annotation-processor mapstruct

javabeans no-reflection

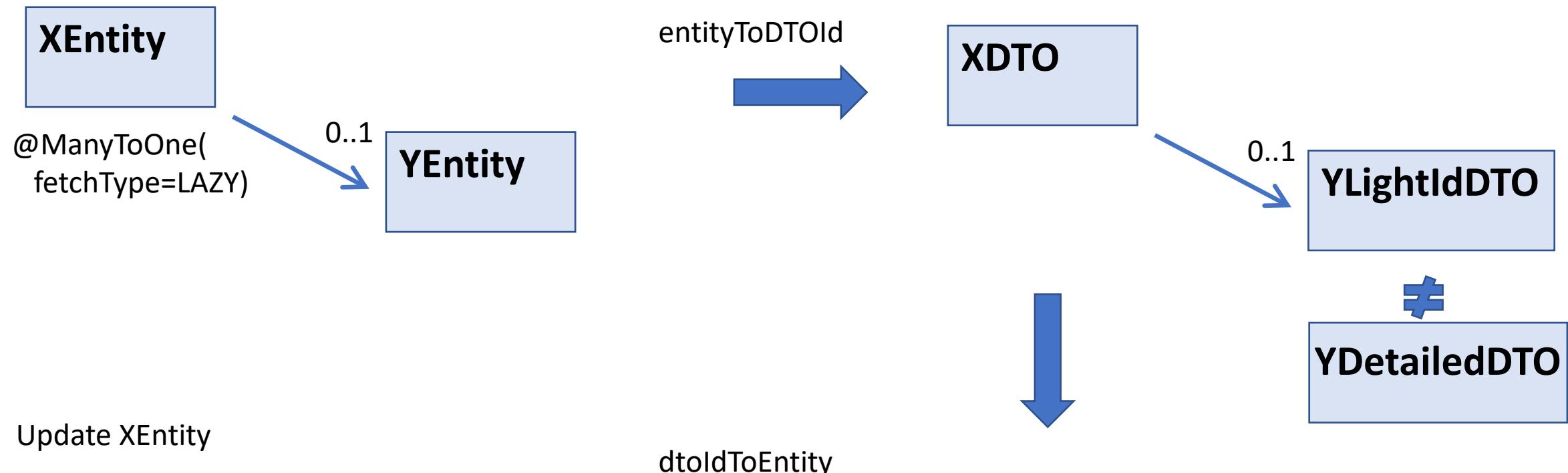
bean-mapping

# Orika ... no findById in Springboot/JPA ?



-- NO built-in recognized support for `findById() « id » <-> « entity »` in JPA + springboot  
NOT suitable for JPA « merge » / need business logic code

# Mapping Entity-DTO + Entity – ID findById !!



Update XEntity

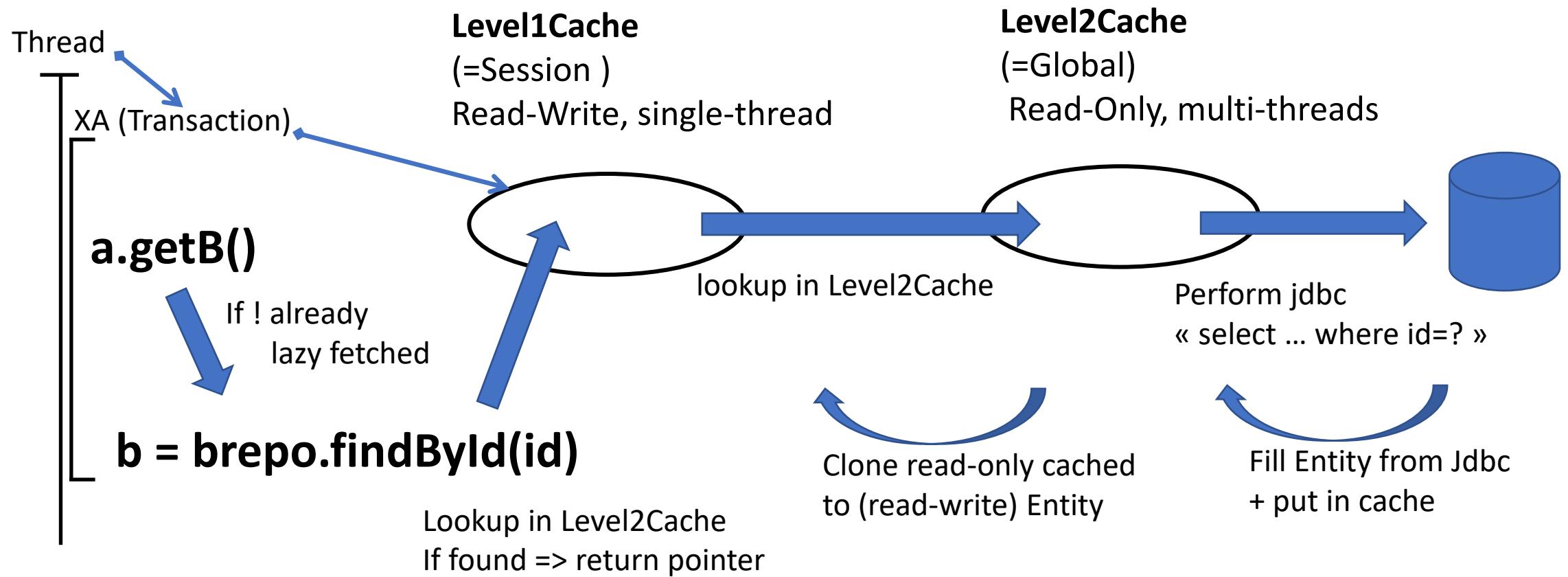
... do not « clean » existing YEntity content !!  
Do not create new empty YEntity !!  
« attach » / « detach » existing Entity by ids

Edit on Client-side ...

- change some fields ( no change ManyToOne )
- Change ManyToOne relation

# Reminder `findById()`

... unique Object in Session per Id



# Buggy code..

## « new Entity(id) + mapper (un-customized) »

```
public BadMergeSourceDTO createBUG(BadMergeSourceDTO src) {  
    BadMergeSourceEntity entity = new BadMergeSourceEntity();  
    dtoConverter.map(src, entity); // copy all mappable fields for dto -> entity ..  
                                // BUG referenceEntity filled EMPTY with Id only .. not merged  
  
    entity = sourceRepo.save(entity);  
  
    sourceRepo.flush(); // .. will be done anyway during commit  
  
    return dtoConverter.map(entity, BadMergeSourceDTO.class);  
                    // => BUG ... invalid DTO return from referenceEntity !!  
}
```

do NOT use « new Entity(id) + mapper (un-customized) »  
need JPA merge? ... use JPA findById() !!

```
public BadMergeSourceDTO create(BadMergeSourceDTO src) {  
    BadMergeSourceEntity entity = new BadMergeSourceEntity();  
    dtoConverter.map(src, entity); // copy all map
```

```
src= BadMergeSourceDTO (id=140)  
  □ field1= null  
  □ field2= null  
  ● id= 0  
  □ ref= BadMergeReferenceIdDTO (id=163)  
    ● displayName= null  
    ● id= 1
```

```
com.example.demo.badmerge.BadMergeSourceDTO@1c39a51
```

```
compareWithRef = referenceRepo.findById(src.getRef().getId())  
compareWithRef= BadMergeReferenceEntity (id=145)  
  □ displayName= "display-name 3" (id=167)  
  □ field1= "field1" (id=172)  
  □ field2= "field2" (id=173)  
  □ field3= "field3" (id=174)  
  □ field4= "field4" (id=175)  
  □ field5= "field5" (id=176)  
  ● id= 3
```



```
.save(entity);  
.map(entity);  
entity= BadMergeSourceEntity (id=142)  
  □ field1= null  
  □ field2= null  
  □ id= 0  
  □ ref= BadMergeReferenceEntity (id=165)  
    □ displayName= null  
    □ field1= null  
    □ field2= null  
    □ field3= null  
    □ field4= null  
    □ field5= null  
    ● id= 1
```

Detached  
(invalid) entity

# ... BUG return invalid after entity2dto ...

```
$ curl -H "content-type: application/json" -H "accept: application/json"      -X POST http://localhost:8080/api/v1/bad-merge -d '{ "ref": { "id": 3 } }'  
{"id":10,"ref": {"id":3,"displayName":null}, "field1":null, "field2":null}
```

DTO from « detached entity »

... invalid (not merged ) => put in session instead of real entity

But value in DB is OK ...

```
$ curl -H "accept: application/json"      http://localhost:8080/api/v1/bad-merge/ref/3  
{"id":3,"displayName": "display-name 3", "field1": "field1", "field2": "field2", "field3": "field3", "field4": "field4", "field5": "field5"}
```

# Sending invalid « id » not in DB ... 500 (in commit) instead of 400 (find in user code)

```
curl -H "content-type: application/json" -H "accept: application/json" \
-X POST http://localhost:8080/api/v1/bad-merge \
-d '{ "ref": { "id": 1 } }' \
{"timestamp":"2022-01-27T16:18:31.484+00:00","status":500,"error":"Internal Server Error","path":"/api/v1/bad-merge"}
```

http 500 ... failed on « server-side » ... should be http 400: bad client input

[org.h2.jdbc.JdbcSQLIntegrityConstraintViolationException](#): Intégrité référentielle violation de contrainte: "FK6LMM3R0UHVWXB5CP8QIUU0EIQ: PUBLIC.BAD\_MERGE\_SOURCE\_ENTITY FOREIGN KEY(REF\_ID) REFERENCES PUB  
Referential integrity constraint violation: "FK6LMM3R0UHVWXB5CP8QIUU0EIQ: PUBLIC.BAD\_MERGE\_SOURCE\_ENTITY FOREIGN KEY(REF\_ID) REFERENCES PUB  
insert into bad\_merge\_source\_entity (field1, field2, ref\_id, id) values (?, ?, ?, ?) [23506-200]  
at org.h2.message.DbException.getJdbcSQLException([DbException.java:459](#)) ~[h2-1.4.200.jar:1.4.200]  
  
at com.zaxxer.hikari.pool.HikariProxyPreparedStatement.executeUpdate(HikariProxyPreparedStatement.java) ~[HikariCP-4.0.3.jar:na]  
at org.hibernate.engine.jdbc.internal.ResultSetReturnImpl.executeUpdate([ResultSetReturnImpl.java:197](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Fin  
at org.hibernate.engine.jdbc.batch.internal.NonBatchingBatch.addToBatch([NonBatchingBatch.java:46](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.persister.entity.AbstractEntityPersister.insert([AbstractEntityPersister.java:3375](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.persister.entity.AbstractEntityPersister.insert([AbstractEntityPersister.java:3908](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.action.internal.EntityInsertAction.execute([EntityInsertAction.java:107](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.engine.spi.ActionQueue.executeActions([ActionQueue.java:604](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.engine.spi.ActionQueue.lambda\$executeActions\$1([ActionQueue.java:478](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at java.util.LinkedHashMap.forEach([LinkedHashMap.java:684](#)) ~[na:1.8.0\_282]  
at org.hibernate.engine.spi.ActionQueue.executeActions([ActionQueue.java:475](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.event.internal.AbstractFlushingEventListener.performExecutions([AbstractFlushingEventListener.java:344](#)) ~[hibernate-core-5.6.1.F  
at org.hibernate.event.internal.DefaultFlushEventListener.onFlush([DefaultFlushEventListener.java:40](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Fina  
at org.hibernate.event.service.internal.EventListenerGroupImpl.fireEventOnEachListener([EventListenerGroupImpl.java:107](#)) ~[hibernate-core-5.6.1.F  
at org.hibernate.internal.SessionImpl.doFlush([SessionImpl.java:1416](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.internal.SessionImpl.managedFlush([SessionImpl.java:507](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.internal.SessionImpl.flushBeforeTransactionCompletion([SessionImpl.java:3299](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.internal.SessionImpl.beforeTransactionCompletion([SessionImpl.java:2434](#)) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
at org.hibernate.engine.jdbc.internal.JdbcCoordinatorImpl.beforeTransactionCompletion([JdbcCoordinatorImpl.java:449](#)) ~[hibernate-core-5.6.1.Final  
at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl.beforeCompletionCallback([JdbcResourceLoc  
at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl.access\\$300\(\[JdbcResourceLocalTransactio  
at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl\\\$TransactionDriverControlImpl.commit\\(\\[Jdbc  
at org.hibernate.engine.transaction.internal.TransactionImpl.commit\\\(\\\[TransactionImpl.java:101\\\]\\\(#\\\)\\\) ~\\\[hibernate-core-5.6.1.Final.jar:5.6.1.Final\\\]  
at org.springframework.orm.jpa.JpaTransactionManager.doCommit\\\(\\\[JpaTransactionManager.java:562\\\]\\\(#\\\)\\\) ~\\\[spring-orm-5.3.13.jar:5.3.13\\\]  
at org.springframework.transaction.support.AbstractPlatformTransactionManager.processCommit\\\(\\\[AbstractPlatformTransactionManager.java:743\\\]\\\(#\\\)\\\) ~\\\[spring\\]\\(#\\)\]\(#\)](#)

# Correct mapper with findById

```
public BadMergeSourceDTO createOk(BadMergeSourceDTO src) {  
    BadMergeSourceEntity entity = new BadMergeSourceEntity();  
    dtoConverter.map(src, entity); // copy all mappable fields for dto -> entity  
  
    BadMergeReferenceIdDTO srcRefDTO = src.getRef();  
    // BUG should be ... refEntity = (srcRefDTO != null)? referenceRepo.getById(srcRefDTO.getId()) : null;  
    // workaround:  
    BadMergeReferenceEntity refEntity = (srcRefDTO != null)? getByIdOrThrow(srcRefDTO.getId()) : null;  
    entity.setRef(refEntity);  
  
    entity = sourceRepo.save(entity);  
    sourceRepo.flush(); // .. will be done after in commit  
  
    return dtoConverter.map(entity, BadMergeSourceDTO.class);  
}  
  
private BadMergeReferenceEntity getByIdOrThrow(long id) {  
    BadMergeReferenceEntity res = // referenceRepo.getById(id);  
        // BUG in hibernate / h2 !!  
        // should throw 400 EntityNotFoundException if invalid input id  
        referenceRepo.findById(id).orElse(null);  
    if (res == null) {  
        throw new EntityNotFoundException("ReferenceEntity not found for invalid input id: " + id);  
    }  
    return res;  
}
```

```
curl -H "content-type: application/json" -H "accept: application/json" \  
-X POST http://localhost:8080/api/v1/bad-merge/create-ok \  
-d '{ "ref": { "id": 3 }}'  
{"id":10,"ref":{"id":3,"displayName":"display-name 3"},"field1":null,"field2":null}
```

# invalid « id » ... bug in Hibernate

## getById => {id=0} BUG!!

## findById => null OK

```
BadMergeReferenceEntity refEntity2 = (srcRefDTO != null)?  
    referenceRepo.findById(srcRefDTO.getId()).orElse(null) : null; // does not throw  
  
BadMergeReferenceEntity refEntity = (srcRefDTO != null)?  
    referenceRepo.getById(srcRefDTO.getId()) // should throw 400 EntityNotFound if invalid input id!!  
    : null;  
entity.setRef(refEntity);  
  
entity = source  
sourceRepo.flu  
  
return dtoConv  
}  
  
Do not trust fields ... hibernate use « ByteBuddyInterceptor » + store in handler
```

# Getting correct 404 for invalid id

```
@RestControllerAdvice
public class RestExceptionHandler extends ResponseEntityExceptionHandler {

    @AllArgsConstructor
    public static class EntityNotFoundErrorResponse {
        public String exceptionType;
        public String errorMessage;
    }

    @ExceptionHandler(EntityNotFoundException.class)
    private ResponseEntity<EntityNotFoundErrorResponse> handleEntityNotFound(EntityNotFoundException ex){
        val error = new EntityNotFoundErrorResponse("EntityNotFoundException", ex.getMessage());
        return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
    }
}

$ curl -H "content-type: application/json" -H "accept: application/json" \
    -X POST http://localhost:8080/api/v1/bad-merge/create-ok \
    -d '{ "ref": { "id": -1000 }}'
{"exceptionType":"EntityNotFoundException","errorMessage":"ReferenceEntity not found for invalid input id: -1000"}
```