

arnaud.nauwynck@gmail.com

Part 5 : Entity to DTO Mapping

Rest Json using JPA ... Entity to DTO, Orika Mapper,
problems with DTO to Entity

This document:

[http://github.com/arnaud-nauwynck/Presentations/java/
Architecture-Design-part5-EntityToDTO.pdf](http://github.com/arnaud-nauwynck/Presentations/java/Architecture-Design-part5-EntityToDTO.pdf)

Exposing DTO + API using Internal Entity

Mapping ... entity2Dto / dto2Entity

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Do not publish

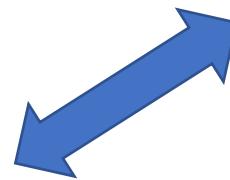
Critical Security data
never publish !!
(Except for owner)

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)



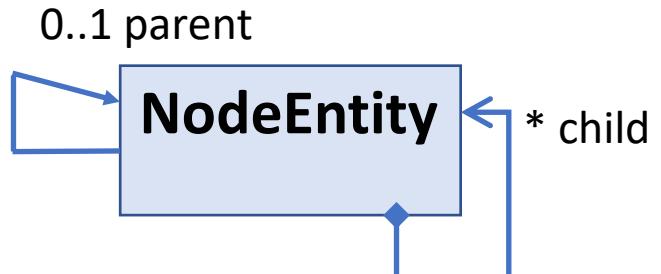
```
@Data  
public class UserLightDTO {  
  
    public String firstName;  
    public String lastName;  
    public String pseudo;  
}
```

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

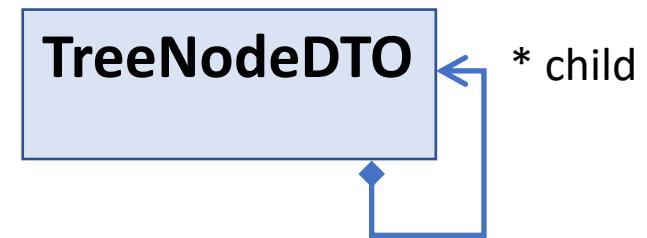
```
@Data  
public class SecuredUserDetailDTO {  
    public String pseudo;  
    public byte[] photo;  
}
```

Entity -> Projection DTOs = « Cutting » relations

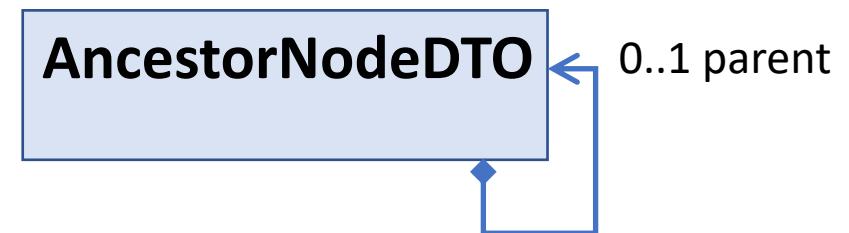
```
@Entity  
@Getter @Setter  
public class NodeEntity {  
  
    @Id  
    private String path;  
  
    // parent -> child relationship  
    @OneToMany(fetch = FetchType.LAZY, mappedBy = "parent")  
    private List<NodeEntity> child;  
  
    // child -> parent (inverse) relationship  
    @ManyToOne(fetch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



Projection Parent->Children (recursive)

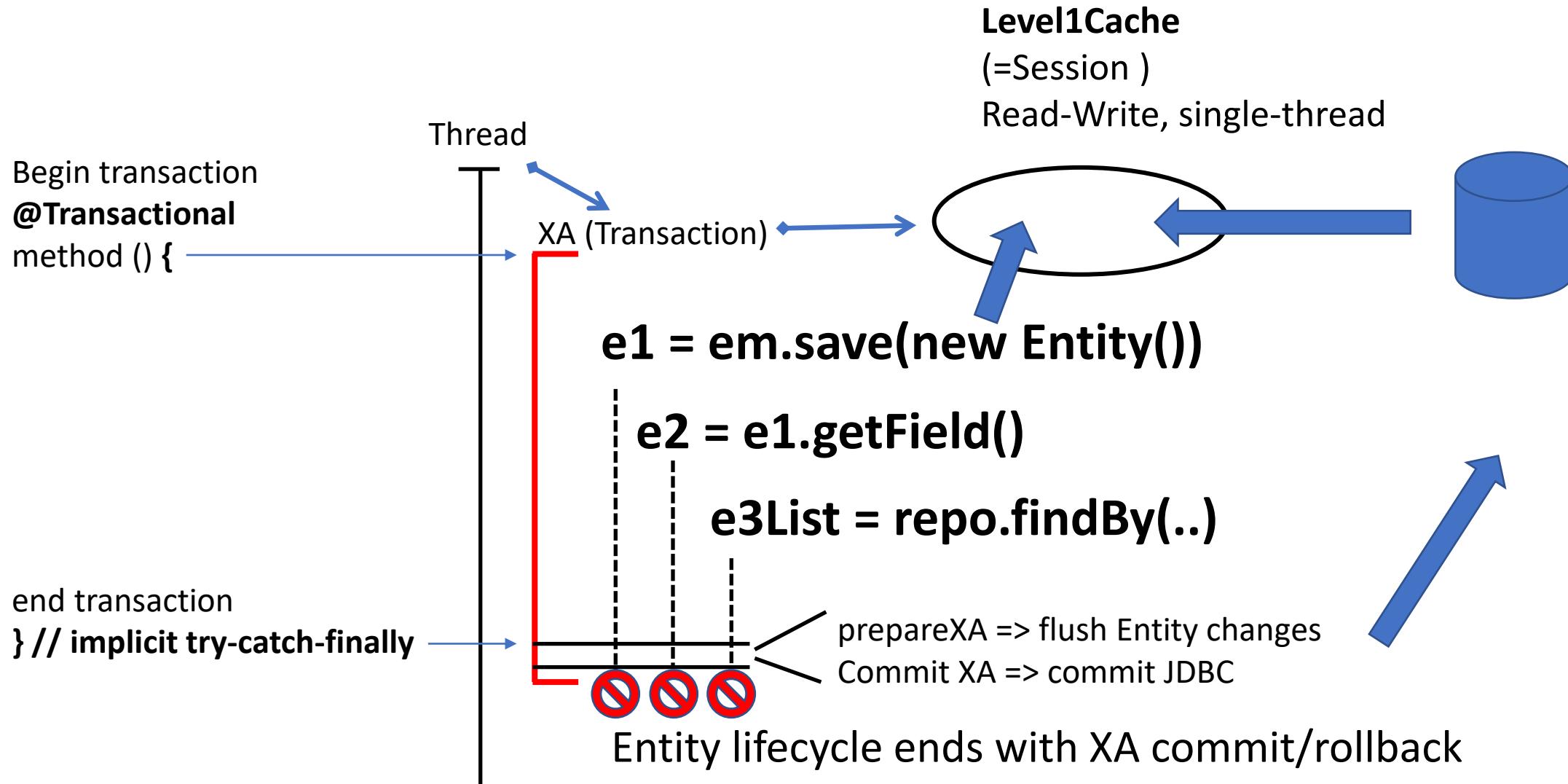


Projection Child->Parent (recursive)

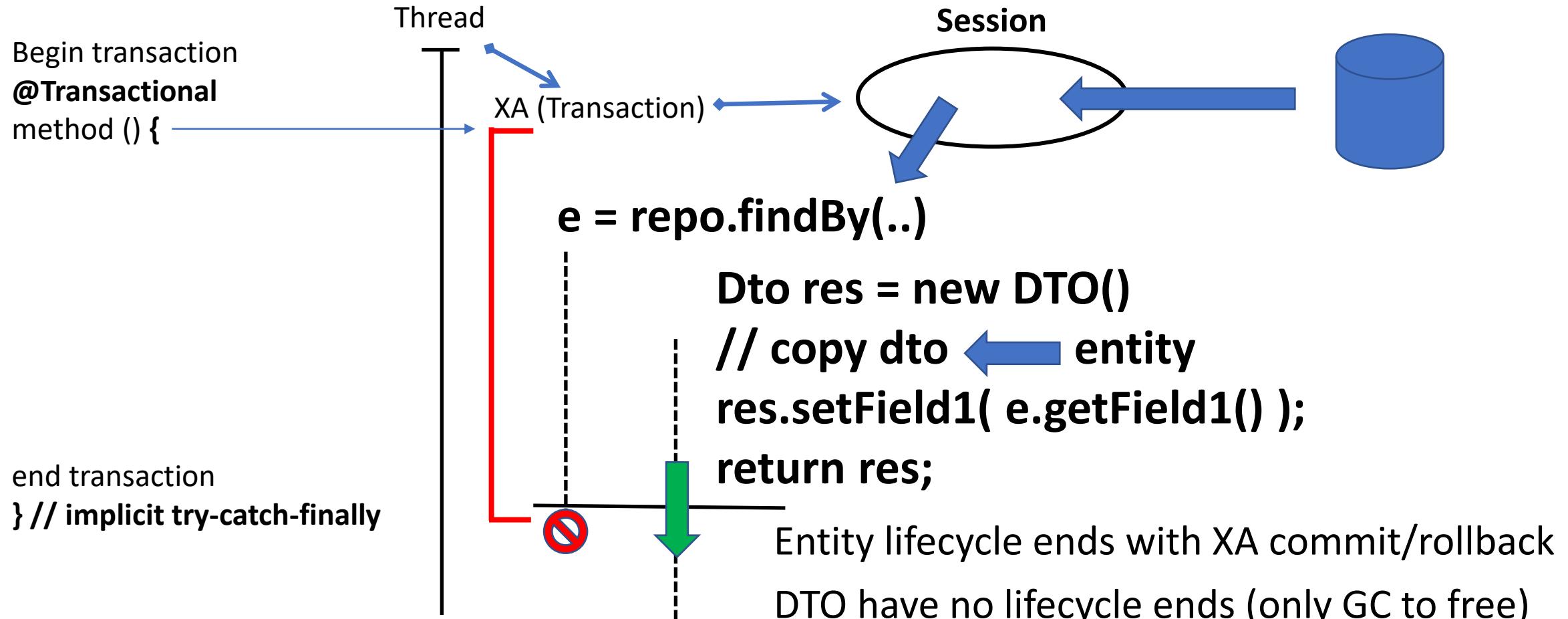


Reminder Part 2 ... Entity lifecycle in Session

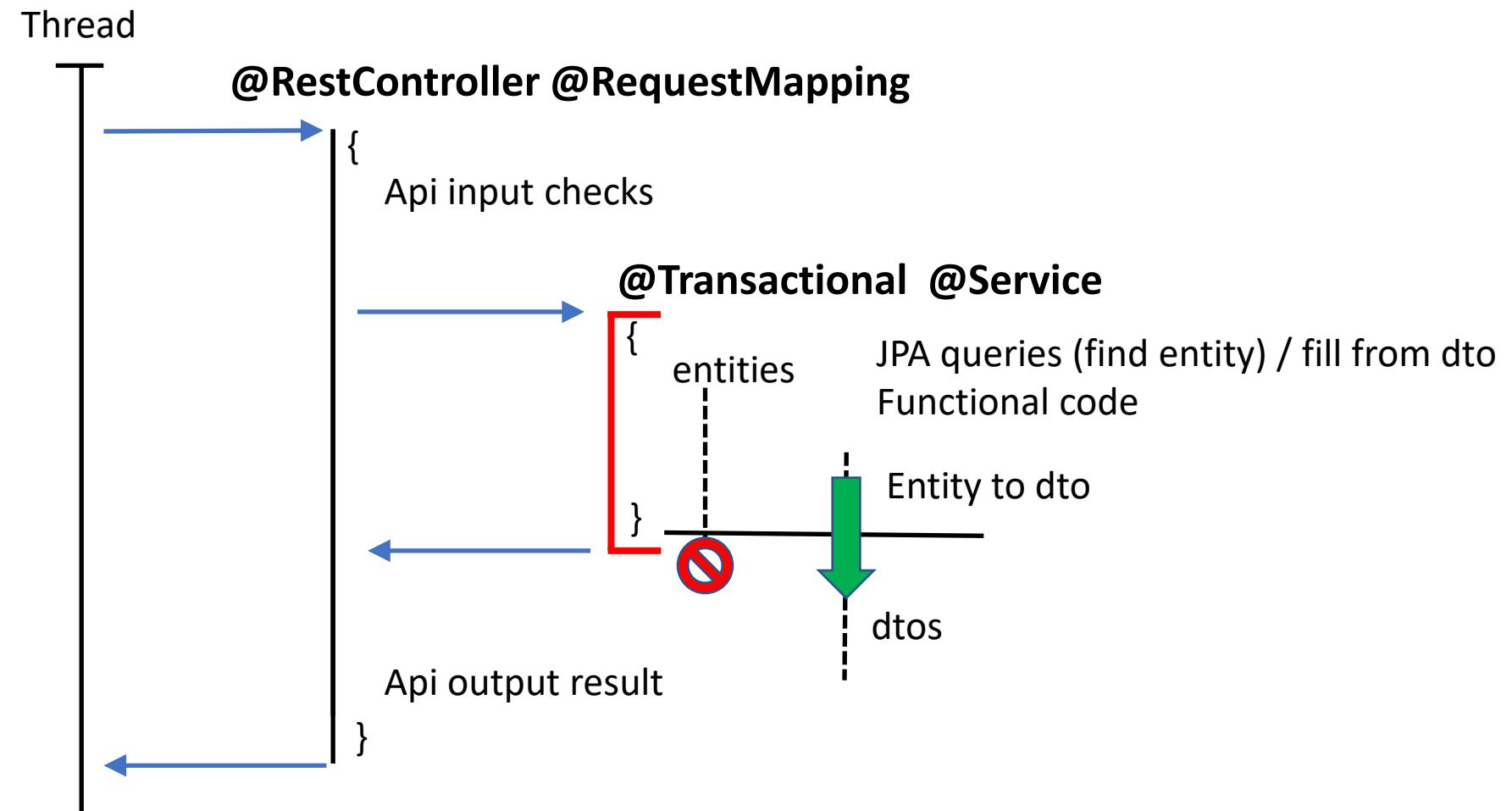
Entity : Lifecycle managed by Session (Transaction)



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

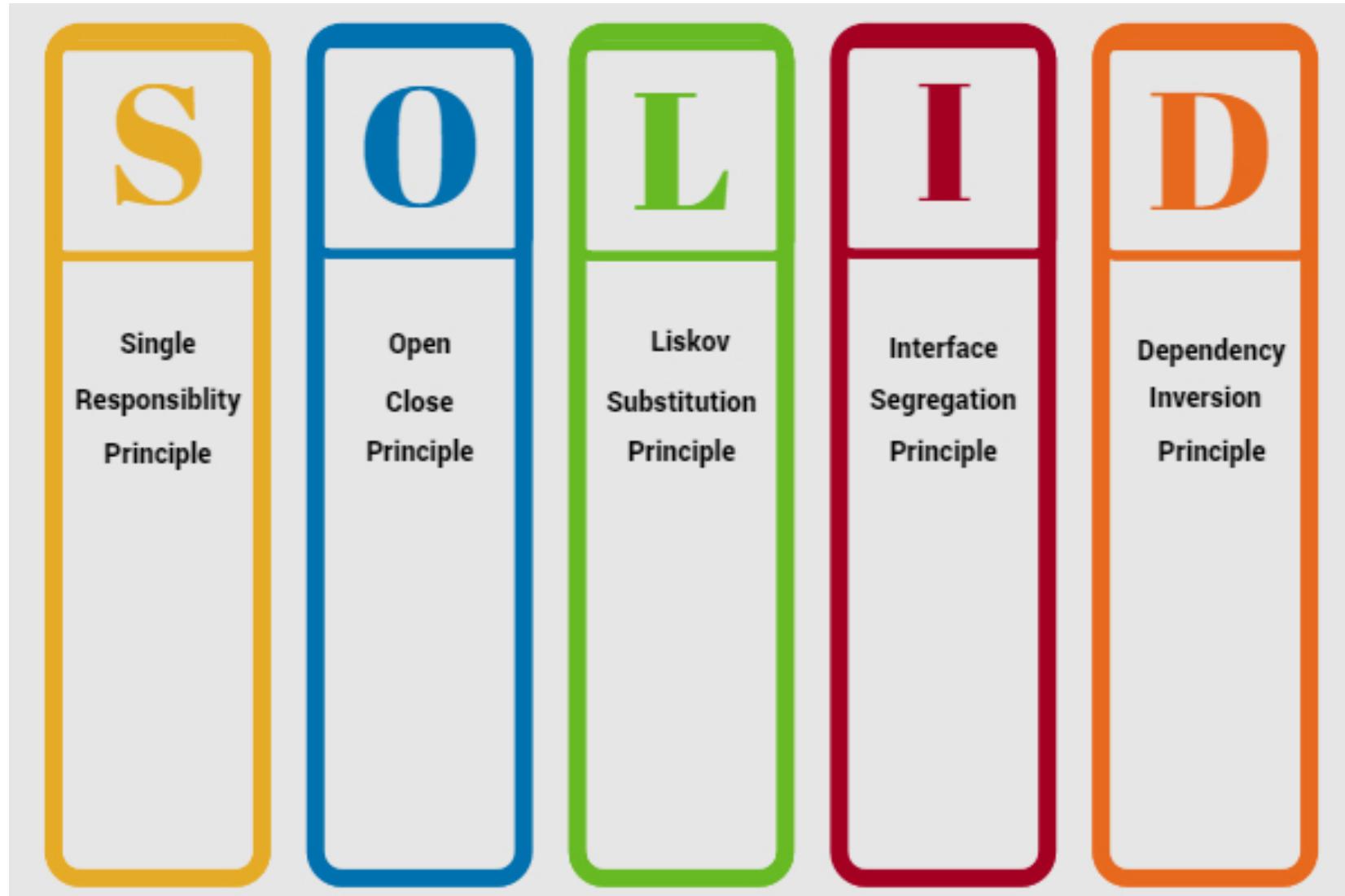
{ unmarshal / delegate / marshall }

```
@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 the GitHub logo, a search bar, and links for Pulls, Issues, Marketplace, and Explore. To the right of the search bar are icons for notifications, a plus sign, and a gear.

The repository name `orika-mapper / orika` is displayed, along with the status "Public". Below it, it says "forked from `elaatifi/orika`".

The main navigation menu includes Code (selected), Issues (141), Pull requests (5), Actions, Projects, Wiki, and a three-dot menu.

Below the menu, there is a dropdown for the branch "master" (selected), a "Go to file" button, an "Add file" button, and a "Code" button (selected). To the right of these buttons is an "About" section.

The "About" section contains the text: "Simpler, better and faster Java bean mapping framework". It also includes a link to `orika-mapper.github.io/orika-docs/`.

A message box indicates that "This branch is 592 commits ahead, 1 commit behind `elaatifi:master`". There is a "Contribute" button next to this message.

The repository's activity feed shows several recent commits:

- `tomashanley-toast` and `elaatifi` Upd... (Oct 19, 2021) - 995 commits
- `core` bump project version to 1.6.0 (4 months ago)
- `orika-janino` bump project version to 1.6.0 (4 months ago)
- `.gitignore` extract Janino into a separate mod... (4 months ago)
- `travis-ci.yml` ignores some tests to fix Linux 17 build (4 months ago)

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

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



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 constraints. 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)
   .field("firstName", "givenName")
   .field("lastName", "sirName")
   .byDefault()
   .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

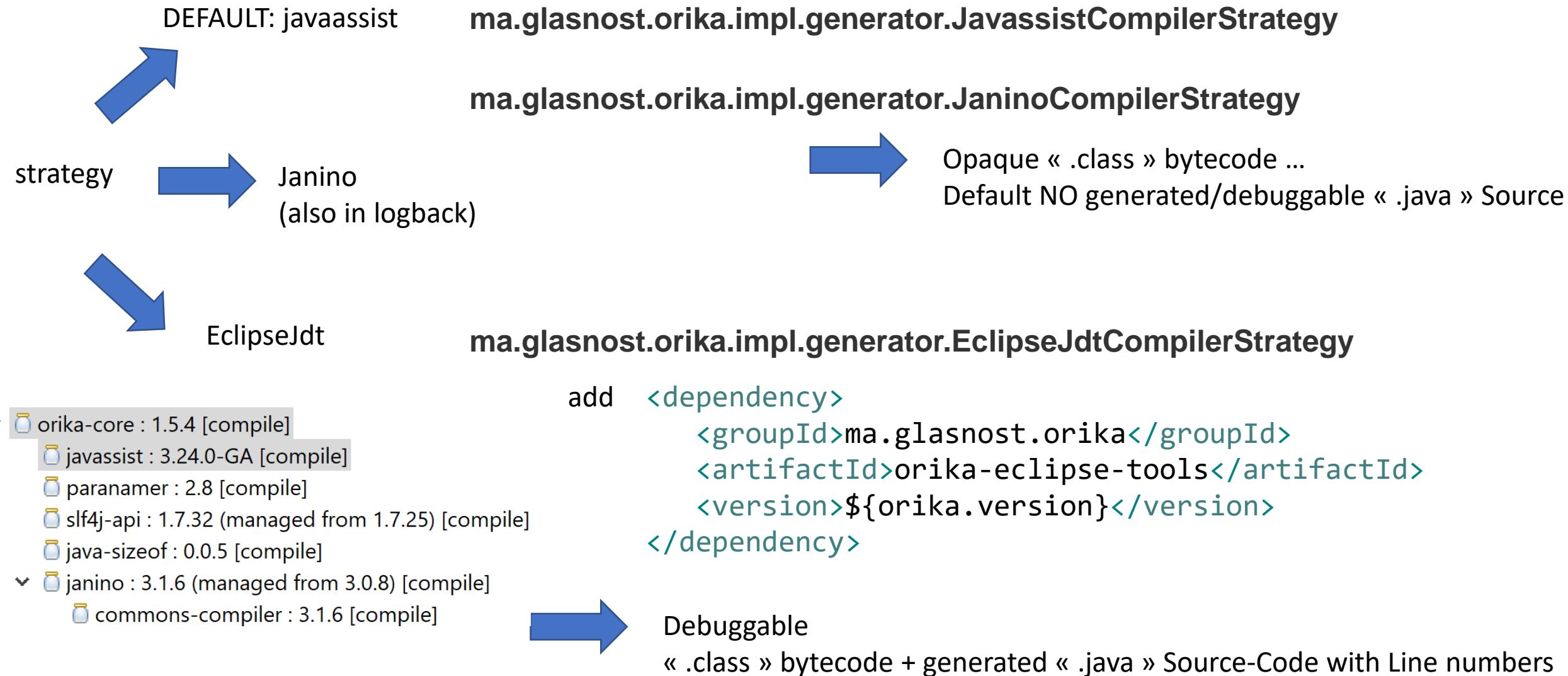
```
/*
 * Specification encapsulates the logic to generate code for mapping comparing a pair of types
 */
public interface Specification {

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

The code editor has a status bar at the bottom with tabs for Console, Problems, Error Log, Debug Shell, Search, Call Hierarchy, and icons for Run, Stop, Refresh, and others.

The output console window below the editor shows the execution of the test and the generated mapper source code:

```
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)

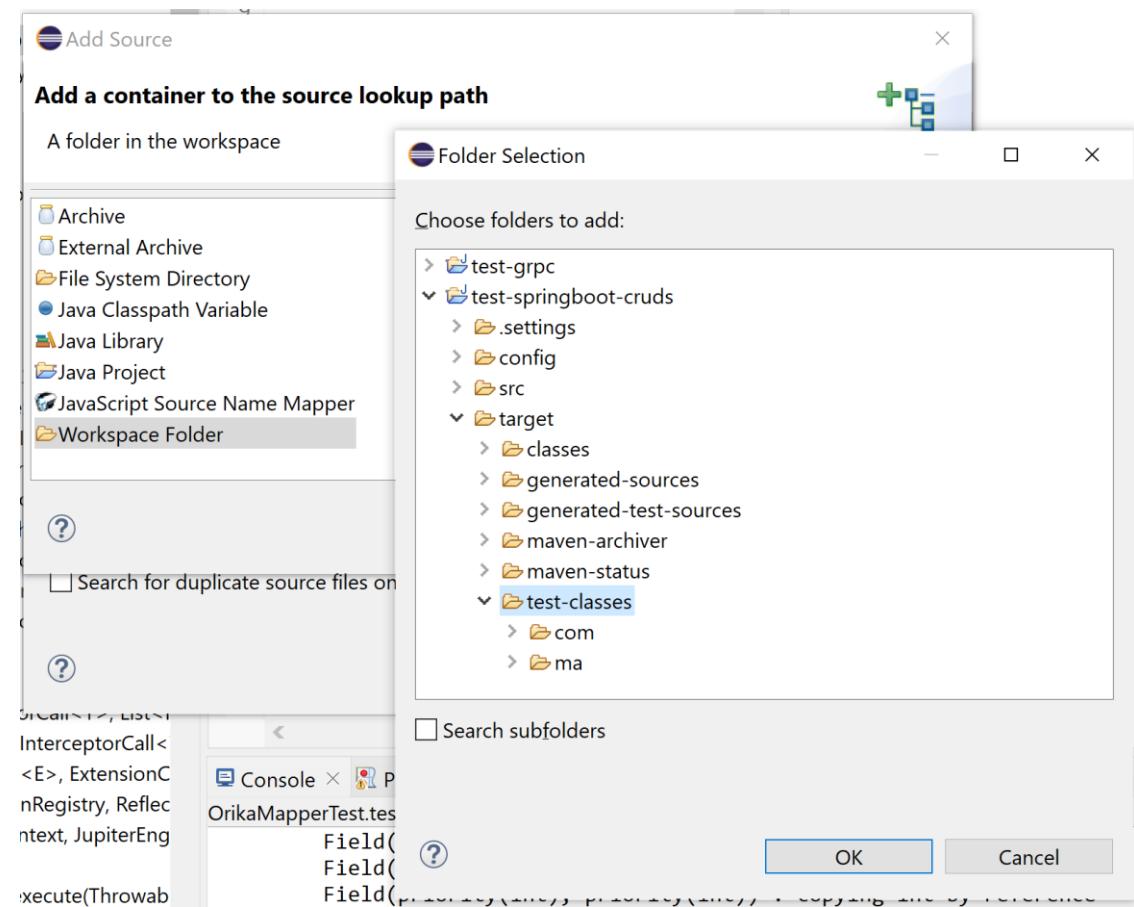
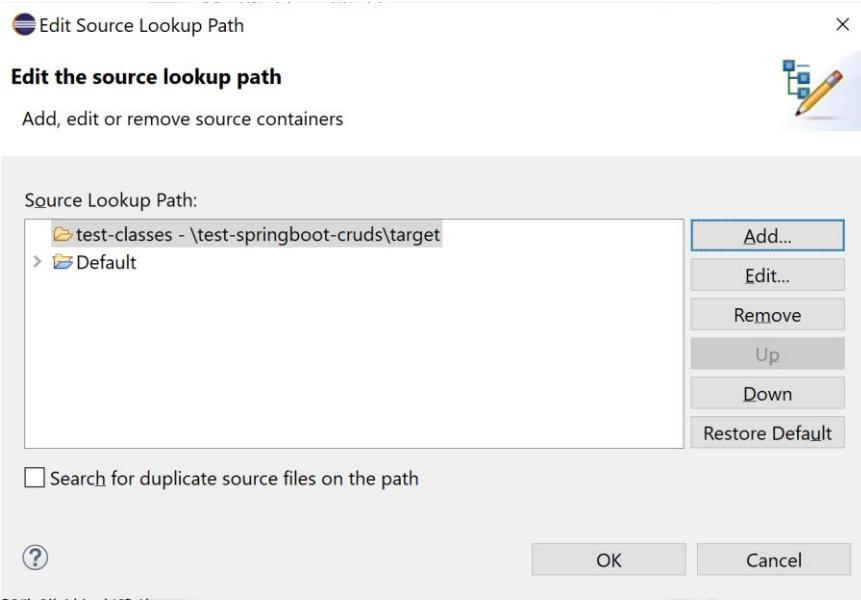
Generated Code .. Source not visible in Eclipse

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 of the `Orika_TodoDTO_TodoEntity_Mapper1027876491952400$0.java` file.

Stack Trace:

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

Source Code (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         sourceMapping.mapAtoB(a, b, sourceMapping.getContext());
7     }
8     super.mapAtoB(a, b, sourceMapping);
9
10    // sourceType: TodoEntity
11    com.example.demo.rest.TodoEntity source = ((com.example.demo.rest.TodoEntity)a);
12    // destinationType: TodoDTO
13    com.example.demo.rest.TodoDTO destination = ((com.example.demo.rest.TodoDTO)b);
14
15    destination.id = ((int)source.getId());
16    destination.label = ((java.lang.String)source.getLabel());
17    destination.priority = ((int)source.getPriority());
18    if(customMapper != null) {
19        customMapper.mapAtoB(source, destination, sourceMapping);
20    }
21 }
22
23
24
25     public void mapBtoA(java.lang.Object a, java.lang.Object b, ma.glasnost.orika.MapperFacadeImpl sourceMapping) {
26         sourceMapping.mapBtoA(a, b, sourceMapping.getContext());
27     }
28     super.mapBtoA(a, b, sourceMapping);
29 }
```

A blue arrow points from a speech bubble containing the text "Supposed to be here « .getLabel() » Line 18 ... ??" to the line number 18 in the code.

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:** Shows tabs for Debug, Project Explorer, Type Hierarchy, JUnit, and several open editor tabs labeled OrikaMapperT..., CodeGenerat..., CompilerStr..., OrikaSystemP..., and Orika_TodoD... .
- Left Side:** Shows the Thread view with a breakpoint at line 21 in TodoEntity.
- Right Side:** Shows the source code for the EclipseJdtCompilerStrategy class. The code is annotated with green circles highlighting specific lines:
 - Line 18: destination.label = ((java.lang.String)source.getLabel());
 - Line 20: if(customMapper != null) {

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 generator / IDE plugins



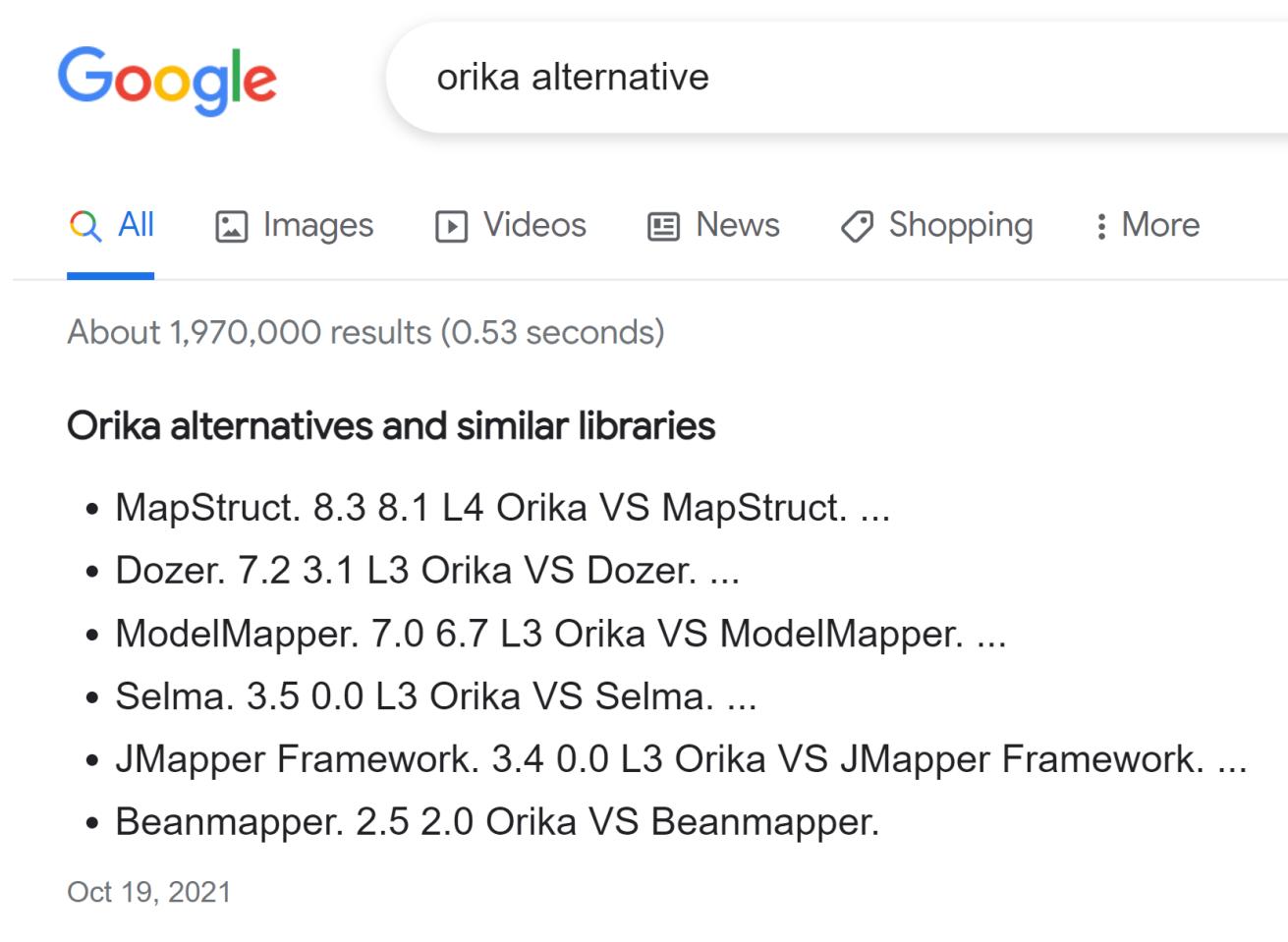
5/ can still see generated code in IDE, on demand



-- 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 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 menu 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, there is a dropdown menu for the `master` branch. In the center, there is 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 it are four more commits: `.github/workflows` (issue `#2591`), `.mvn/wrapper`, `build-config`, and `core-jdk8`. On the right side, there is a section titled `About` with the description "An annotation processor for generating type-safe bean mappers". Below this is a link to `mapstruct.org/`. At the bottom, there is a list of tags: `java`, `mapping`, `annotation-processor`, `mapstruct`, `javabeans`, `no-reflection`, and `bean-mapping`.

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

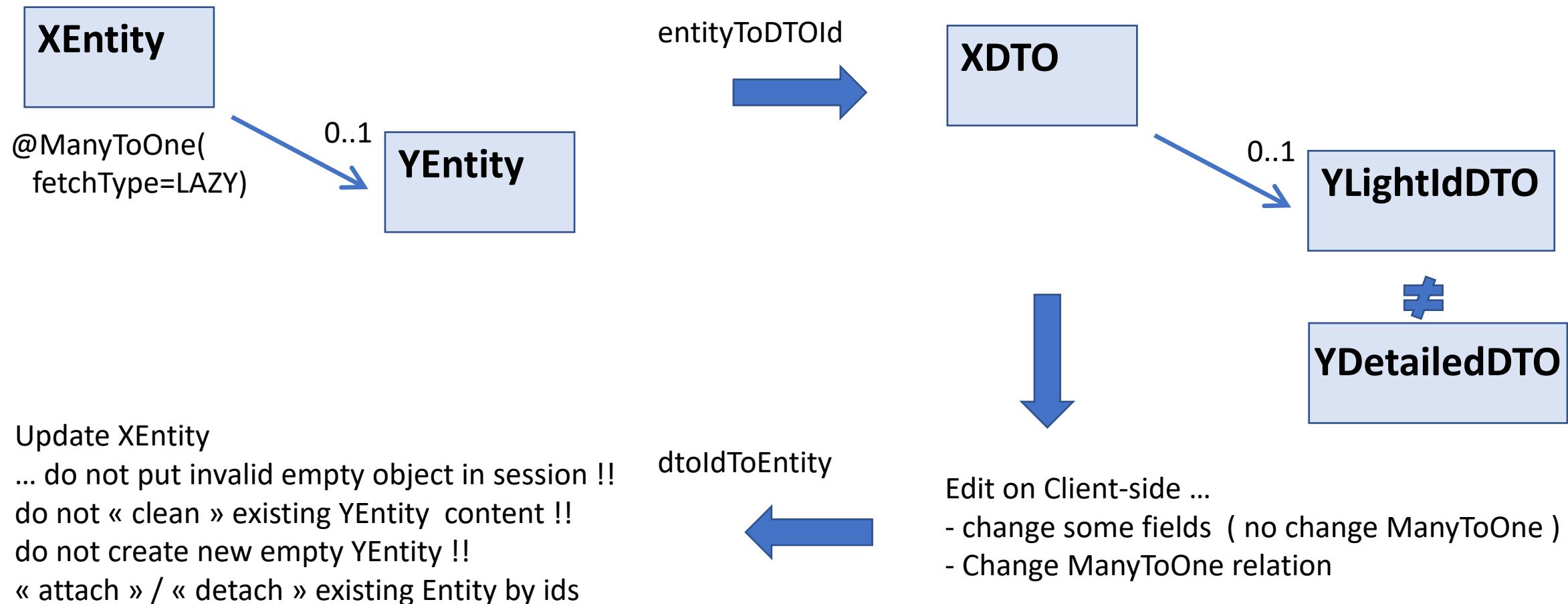
core-jdk8 [maven-release-plugin] prepare for next ... 2 months ago

Orika ... no findById in Springboot/JPA ?



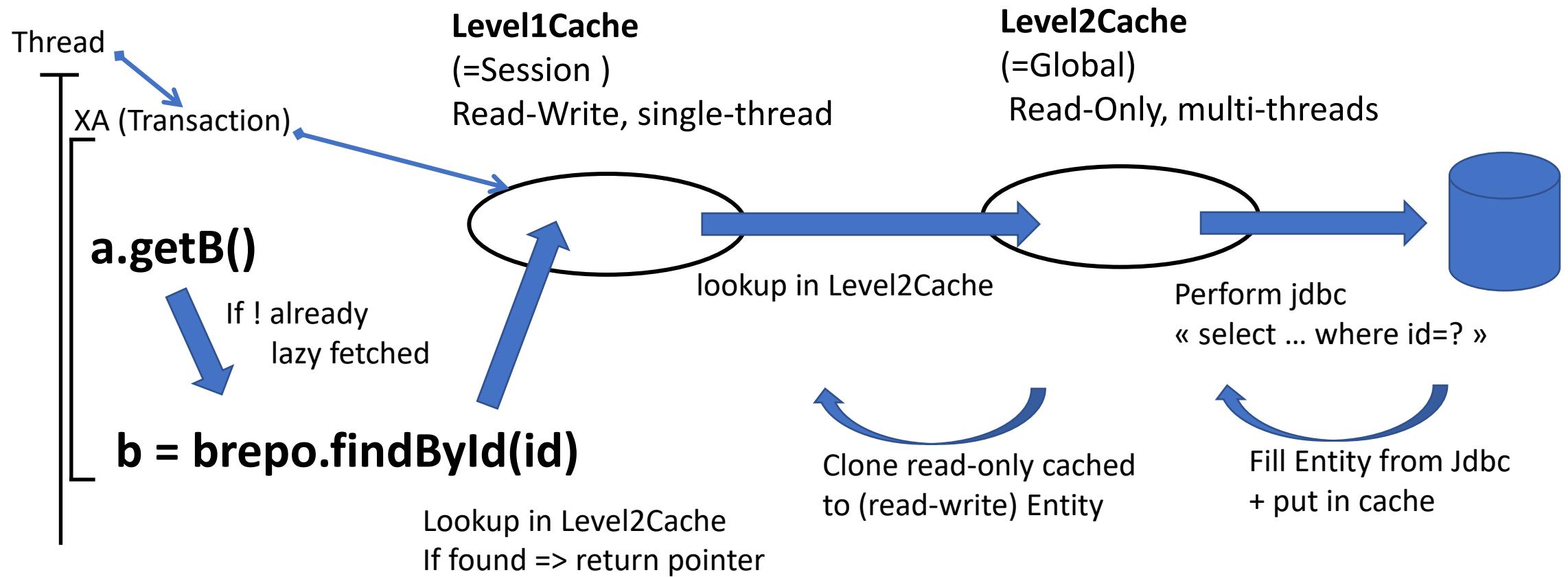
-- NO built-in recognized support for `findById() « id » <-> « entity »` in JPA + springboot
NOT suitable for JPA « merge » / need business logic code

Mapping Entity-DTO + Entity – ID findById !!



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() !!

The screenshot illustrates a comparison between two code snippets and their execution results in an IDE.

Code Snippet 1:

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

Execution Result 1 (Left):

Variables:

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

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

Code Snippet 2:

```
compareWithRef = referenceRepo.findById(src.getRef().getId())
```

Execution Result 2 (Bottom Left):

Variables:

- 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

Comparison Result:

A large blue ≠ symbol is centered between the two execution results.

Execution Result 3 (Bottom Right):

.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

com.example.demo.badmerge.BadMergeSourceEntity@ada730

Annotation:

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"}
```

Save + setRef ... without save(ref) (no « Cascade Create »)

```
Caused by: org.hibernate.TransientPropertyValueException: object references an unsaved transient instance - save the transient instance before flushing :  
    at org.hibernate.engine.spi.CascadingActions$8.noCascade(CascadingActions.java:379) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.engine.internal.Cascade.cascade(Cascade.java:169) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.event.internal.AbstractFlushingEventListener.cascadeOnFlush(AbstractFlushingEventListener.java:159) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.event.internal.AbstractFlushingEventListener.prepareEntityFlushes(AbstractFlushingEventListener.java:149) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.event.internal.AbstractFlushingEventListener.flushEverythingToExecutions(AbstractFlushingEventListener.java:82) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.event.internal.DefaultFlushEventListener.onFlush(DefaultFlushEventListener.java:39) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.event.service.internal.EventListenerGroupImpl.fireEventOnEachListener(EventListenerGroupImpl.java:107) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
    at org.hibernate.internal.SessionImpl.doFlush(SessionImpl.java:1416) ~[hibernate-core-5.6.1.Final.jar:5.6.1.Final]  
... 44 common frames omitted
```

Mapper + Hibernate + Transaction



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

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



Hibernate is a MESS

BUG : put invalid « un-merge » Entity in session

BUG : no EntityNotFoundException error code

Can not see field values at debug type, use ByteBuddyInterceptors ... duplicate memory needs



Switch to EclipseLink (reference implementation of JPA)

More problems...

Hard-coded DTO API to chose field projections (and cut other fields)

1 rule does not fit all clients

DTO fields computation + transfer

- :(For some clients : not enough fields in DTO ... need more
=> client loop many http calls
- :(For some clients : too many (useless) fields in DTO ... need less
=> slow on server to compute too many JDBC selects,
and transferring useless network data

GraphQL to the rescue ..



Switch to

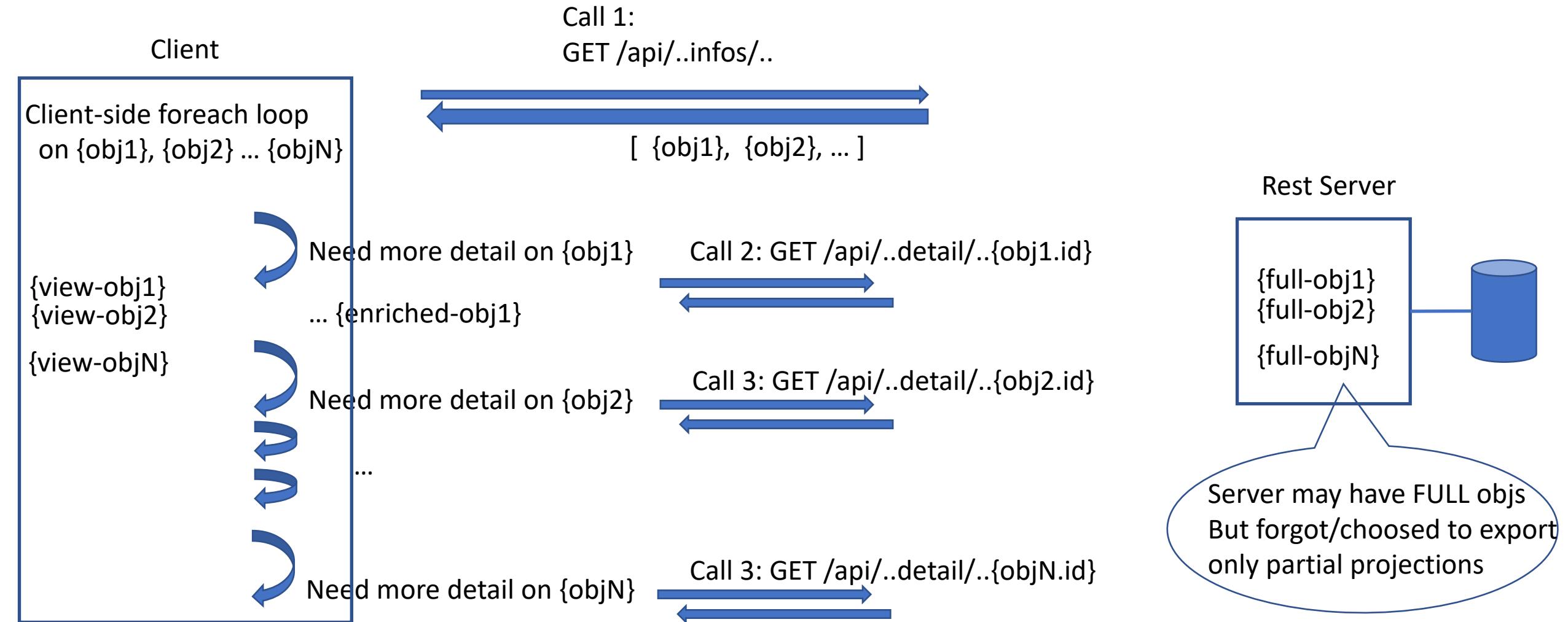


GraphQL

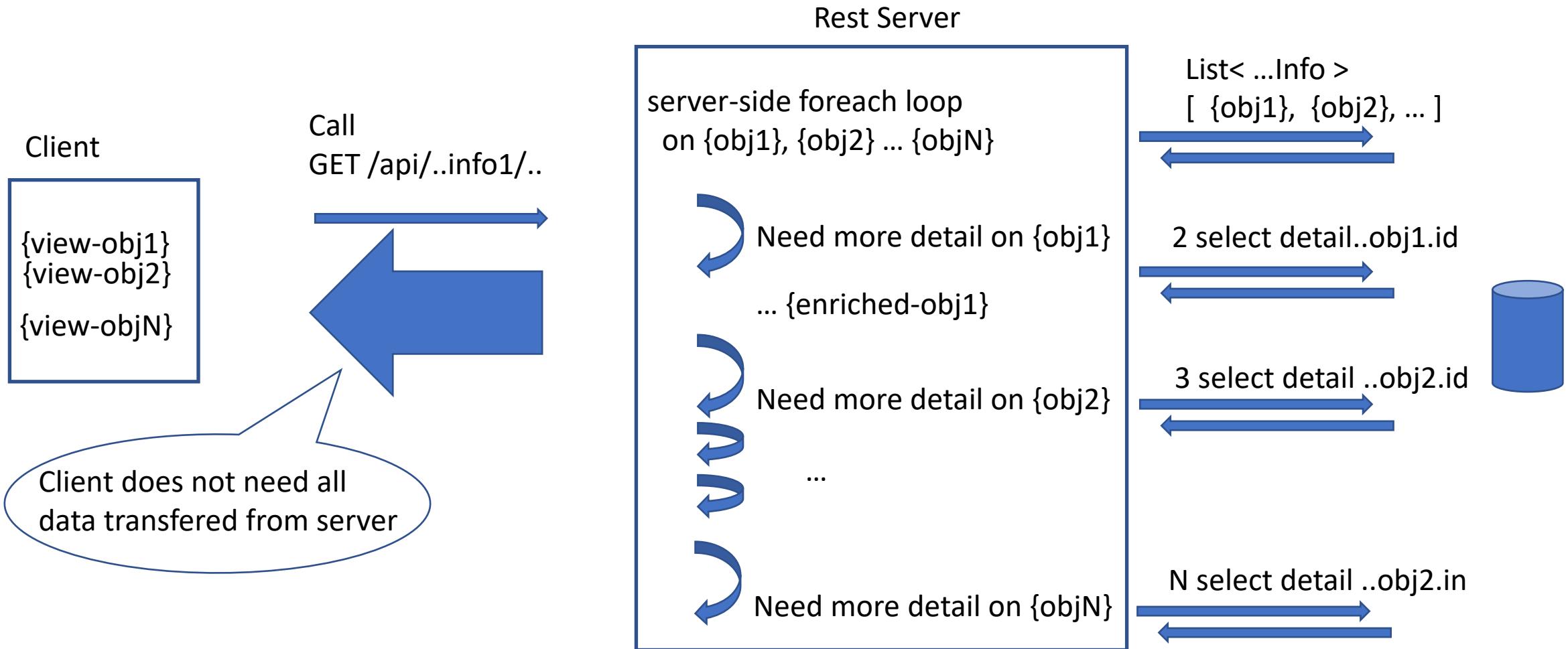
java + annotation + springboot ecosystem for GraphQL

Much more powerfull & simpler than DTO + Mappers !!

Not enough fields in Api DTO client loop to enrich.. 1+N network calls



Too Many fields in (useless) detailed DTO ... server-side loop 1+N JDBC requests



By-Passing « Query » « Entity » then to « DTO »
... direct Query with JOINs to DTOs

Every application has a central « Search screen »

Such API must be highly optimized

Worth for back to the basics: « JPA QL » or plain old « Jdbc »

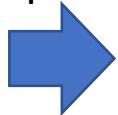
Do not optimize all apis in JDBC for maintenance/evolutivity

Direct JPA QL to DTO (or even Native SQL to DTO)

```
@Autowired
private EntityManager em;

public List<CustomQueryResultDTO> query1(String param) {
    String jpaQL = "SELECT new com.example.demo.directquery2dto.CustomQueryResultDTO(" //
        + " p.id, p.field1, p.field2, " //
        + " p.ref.id, p.ref.field1, p.ref.field2 " // <= field navigation, equivalent to Sql tables join
        + " )" //
        + " FROM DirectSourceEntity p " //
        + " WHERE p.field1 LIKE :param"; //
    TypedQuery<CustomQueryResultDTO> q = em.createQuery(jpaQL, CustomQueryResultDTO.class);
    q.setParameter("param", param);
    List<CustomQueryResultDTO> resDtos = q.getResultList();
    return resDtos;
}
```

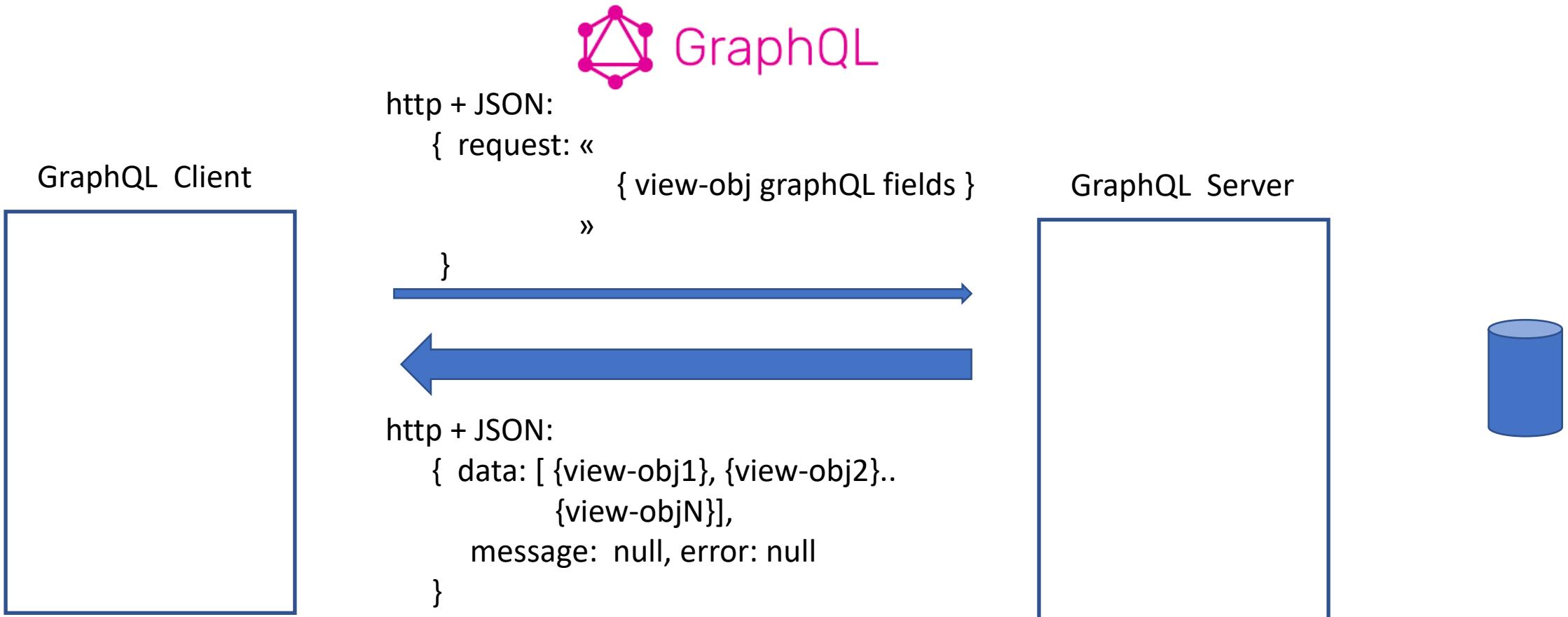
1 SQL request



```
SELECT
    p.id, p.field1, p.field2, p.ref_id, ref.field1, ref.field2
FROM direct_source_entity p
CROSS JOIN direct_reference_entity ref
WHERE p.ref_id=ref_.id
and p.field1 like ?
```

GraphQL

specific query on client – Answer exact from Server



GraphQL Main Features & Benefits...

- **RootQuery** ... ~ http GET + swagger
 - Strongly Typed schema + documented
 - discovery by introspection (metadata query)
 - Interactive Graphiql browser
- **Mutations** ... ~ http POST/PUT/DELETE actions
- **Subscriptions** ... ~ Web socket to subscribe



SIMPLER than DTO + Mapper + Projections/Cuts dilemma



& BEST evolutivity



& BETTER performances



NO pre-defined « all-in-one client data/prepared calls »

... See next lessons

Conclusions

Modeling Domain Entity / Optimizing Database was « essential & difficult »

Standard JSON marshalling/unmarshalling is done via DTO classes
and entity2dto / dto2entity mapper

Mapper in way « dto2entity » need complex JPA / Transactional code
(findById / save ... otherwise strange bugs appear)

Exposing API with specific DTOs (choosing projections/cuts)
looks « accidentally difficult » GraphQL to the rescue