arnaud.nauwynck@gmail.com

Architecture Design

Part 3 : Api, DTO

external APIs, Protocol Marshalling, Entity to DTO, Mapper

This document:
http://github.com/arnaud-nauwynck/Presentations/java/Architecture-Design-part3-DTO.pdf

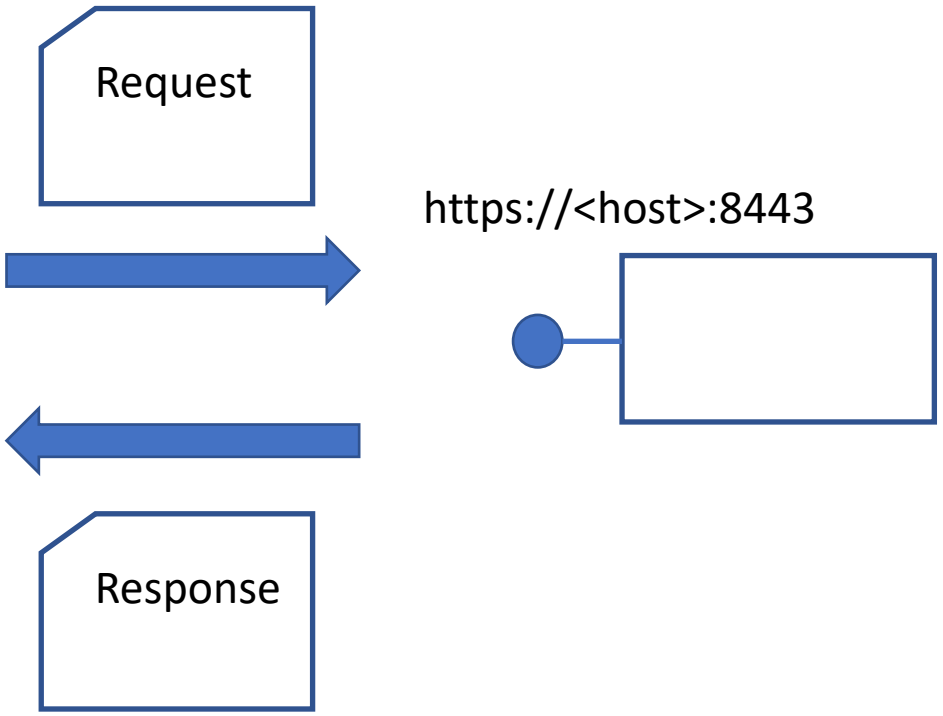
Api http Server … for http Clients

curl://

POSTMAN

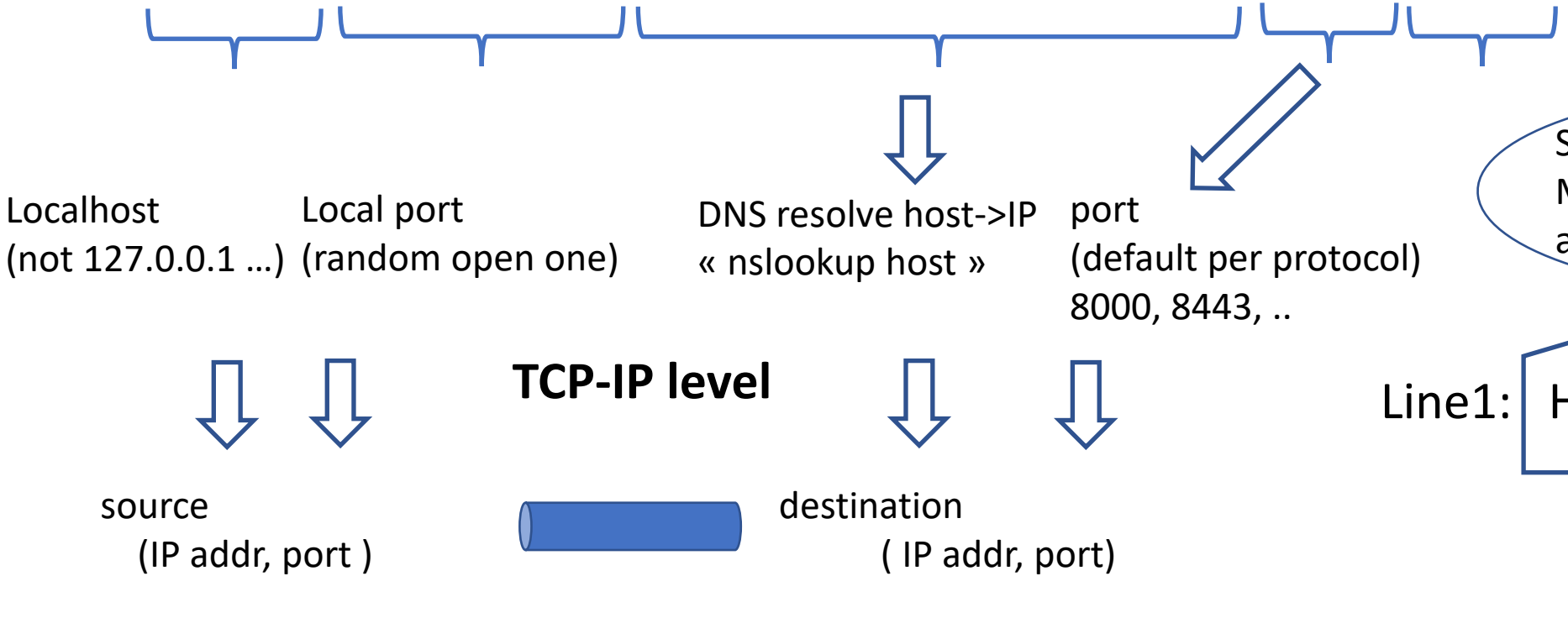
swagger

Retrofit



http Call: VERB + path + headers + body
« URL » = host + port + Path

https://user:pass@somehost.some.domain:8443/path

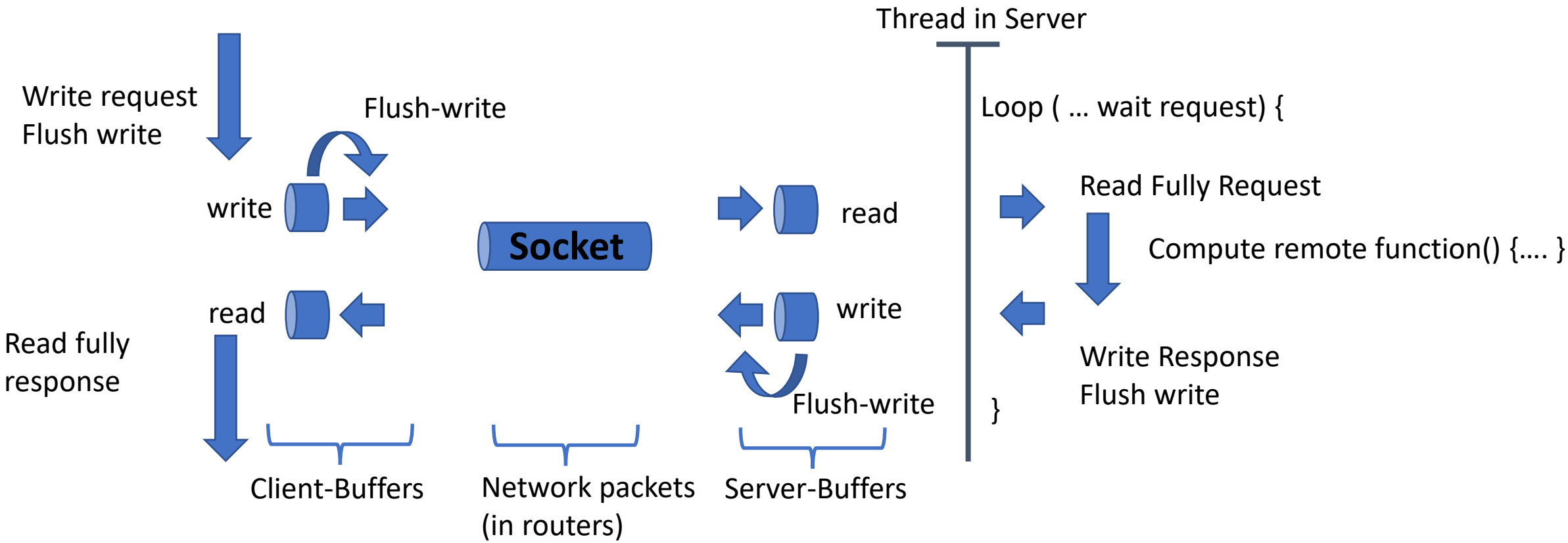


Socket = quadruplet (srcIP,srcPort,destIP,destPort)

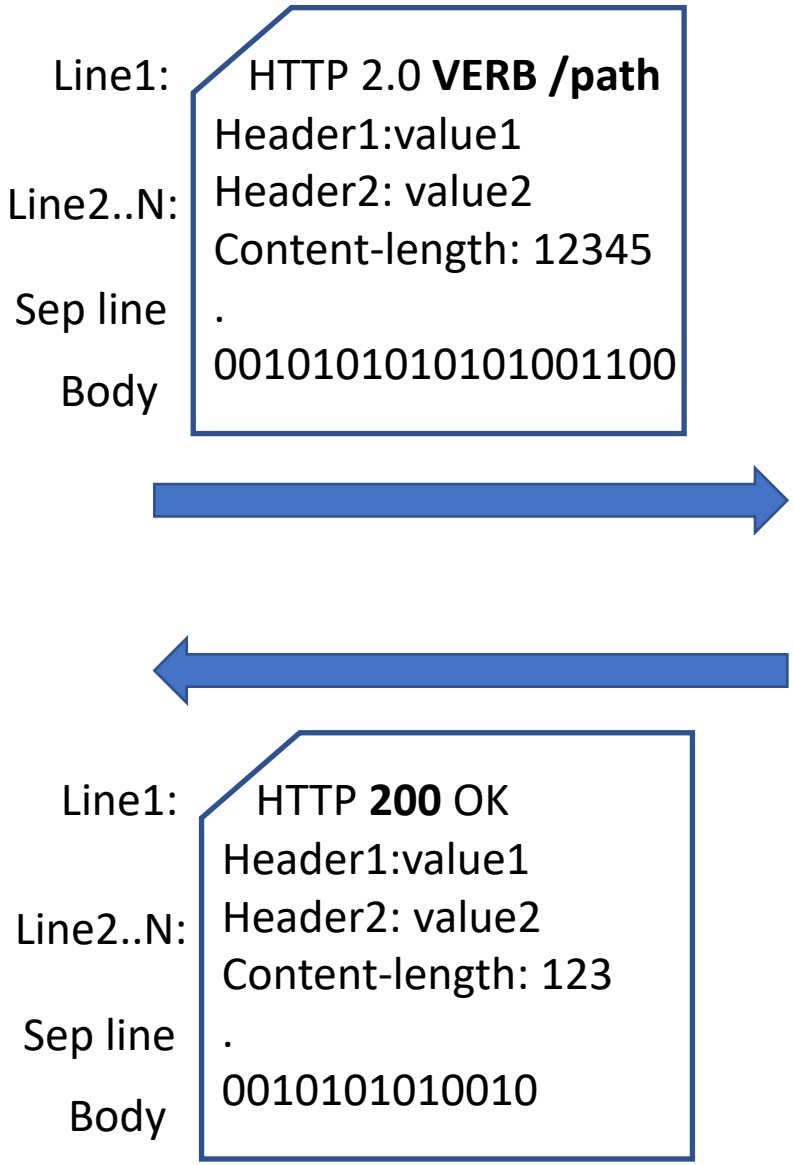
Remote « Call »
asymetric socket read-write
active Client – waiting Server, Processing

On Client-side:
Write,flush-write … then (blocking-wait) read

On Server:
(blocking-wait) read … then write,flush-write



Simple Http Request Model



Rest (http + Json) Requests

Verb: GET / POST / PUT / DELETE / PATCH / HEAD

Line1: HTTP 2.0 **VERB /resource-path**
Line2..N: Header1:value1
Header2: value2
Content-length: 12345
Sep line .
Body **{ json }**

Path correspond to Resource « id » (maybe with a parent/child hierarchy)

GET = read-only (so repeatable may be cached)

POST = create (or submit request) not idem-potent! warn: do not resubmit if OK

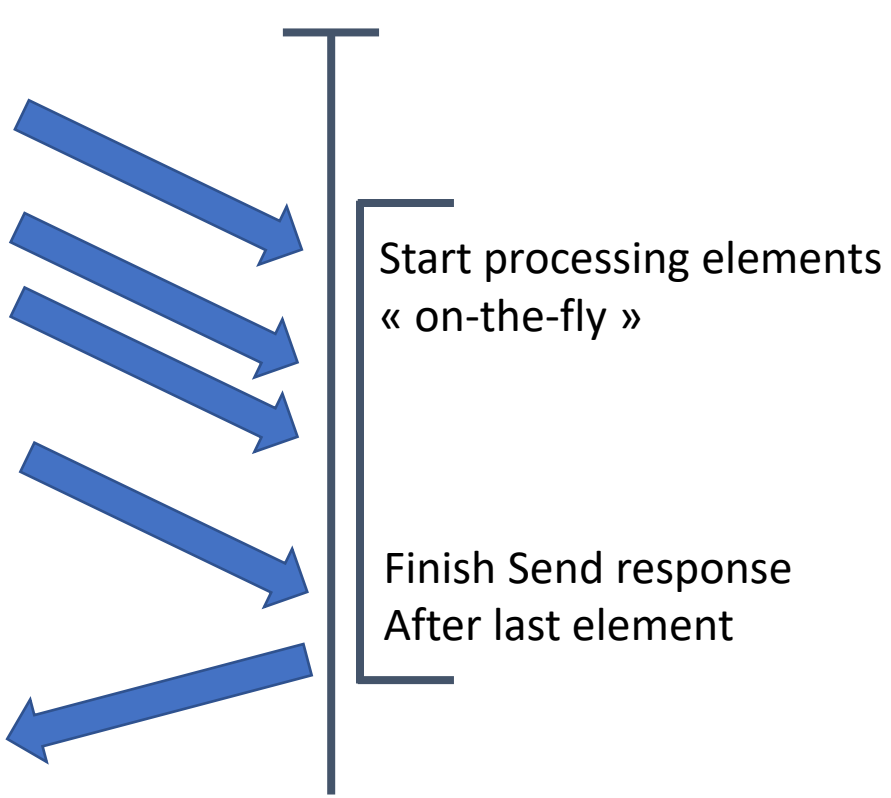
Request/Resource data encoded in JSON

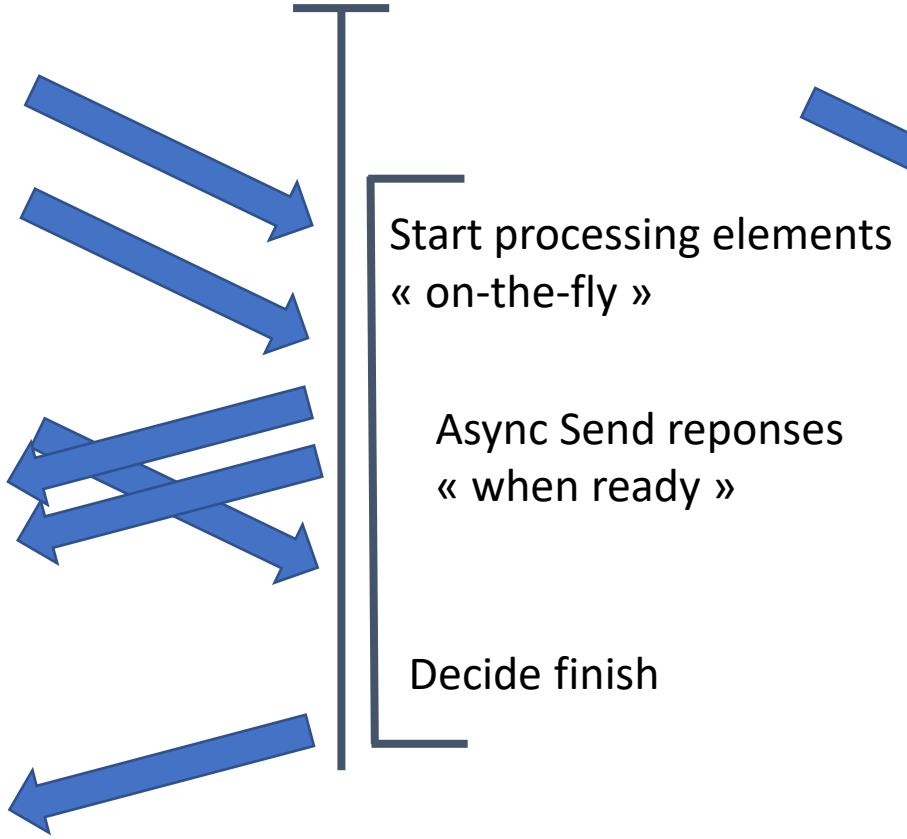
PUT = update/upsert (idem-potent) can be resubmitted

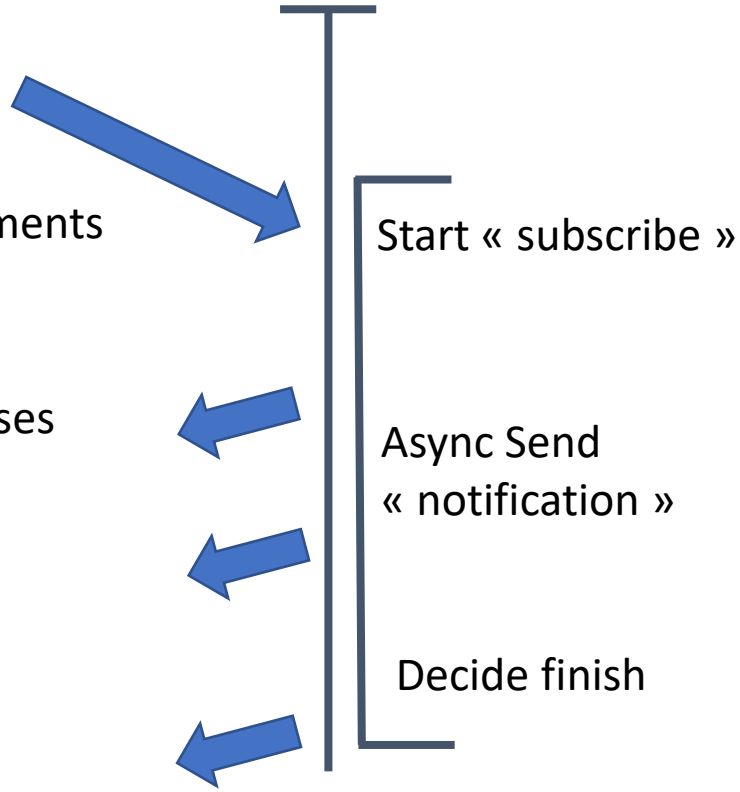
DELETE = (idem-potent) can be resubmitted

Line1: HTTP **200** OK
Line2..N: Header1:value1
Header2: value2
Content-length: 123
Sep line .
Body **{ json }**

Other Asynchronous Call Models





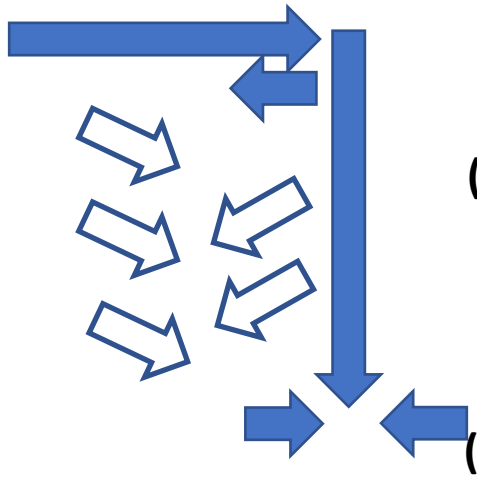


Example gRPC (over Https..)

```
service Greeter {

  // send 1 request => receive 1 response
  rpc SayHello (HelloRequest) returns (HelloReply);

  // send 1 request => receive * responses stream
  rpc SayHello_OutStream (HelloRequest) returns (stream HelloReply);

  // send * request stream => 1 response  (wait X request, then respond)
  rpc SayHello_InStream (stream HelloRequest) returns (HelloReply);

  // send * request stream => receive * responses stream
  rpc SayHello_InStream_OutStream (stream HelloRequest) returns (stream HelloReply);

}

message HelloRequest {
  string name = 1;
}

message HelloReply {
  string message = 1;
}
```

WebSocket
(ws:// instead of https://)

ws://somehost.somedomain



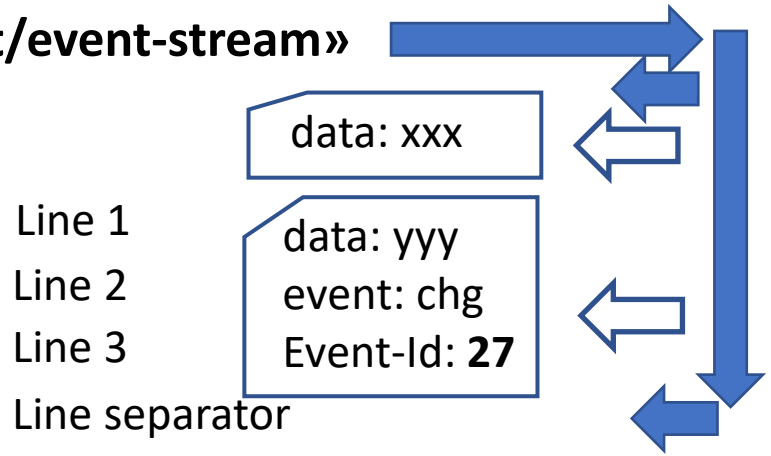
(=> TCP-IP socket kept opened …)

(=> TCP-IP socket closed by client / server)

SSE : Server – Sent – Event
+ still https:// with socket negotiation
« medium wait » Polling by client

http GET /sse

Accept: « text/event-stream»



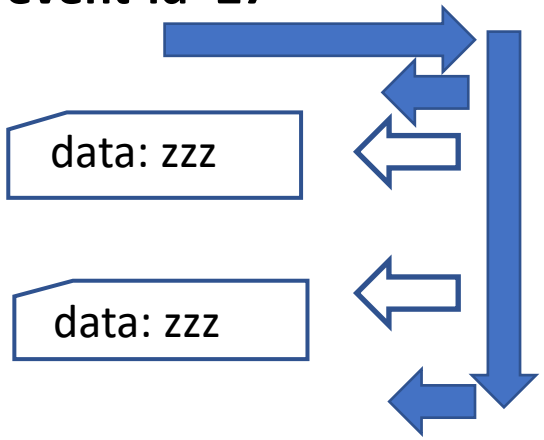
HTTP 200 OK

Content-type: **text/event-stream**

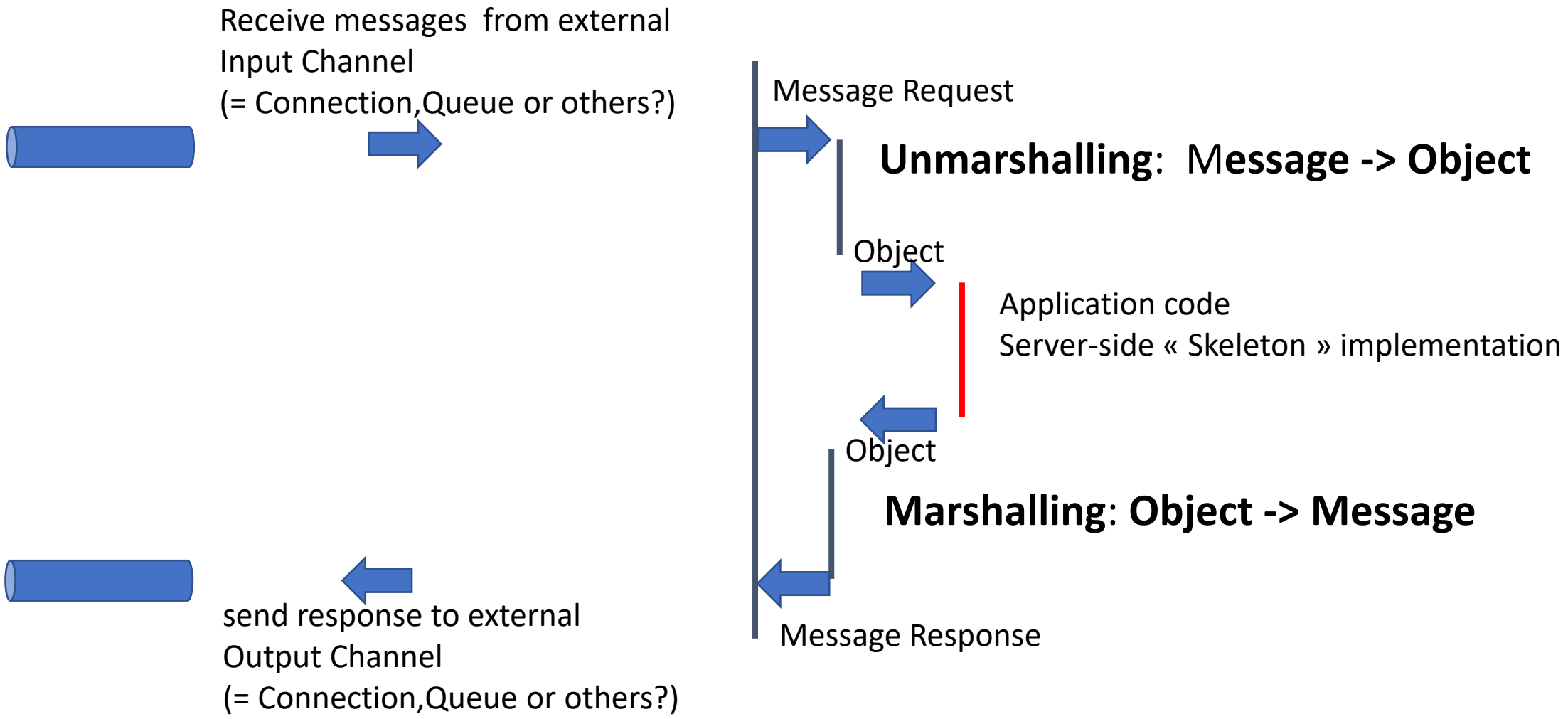
(=> socket kept opened …)

Timeout … response / close connection

http GET /sse?**last-event-id=27**



External Protocol
… Marshalling/Unmarshalling to Internal Langage



Message Encoding : Text / Binary

explained

verbose

compressed

opaque

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

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

<xml?> <a> some value

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

Simplest
Compromise for Web

CSV: a.b; \n
some value

Binary: 01010101011010101010

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

Schema … Code-Generator

Text
(No code generator)

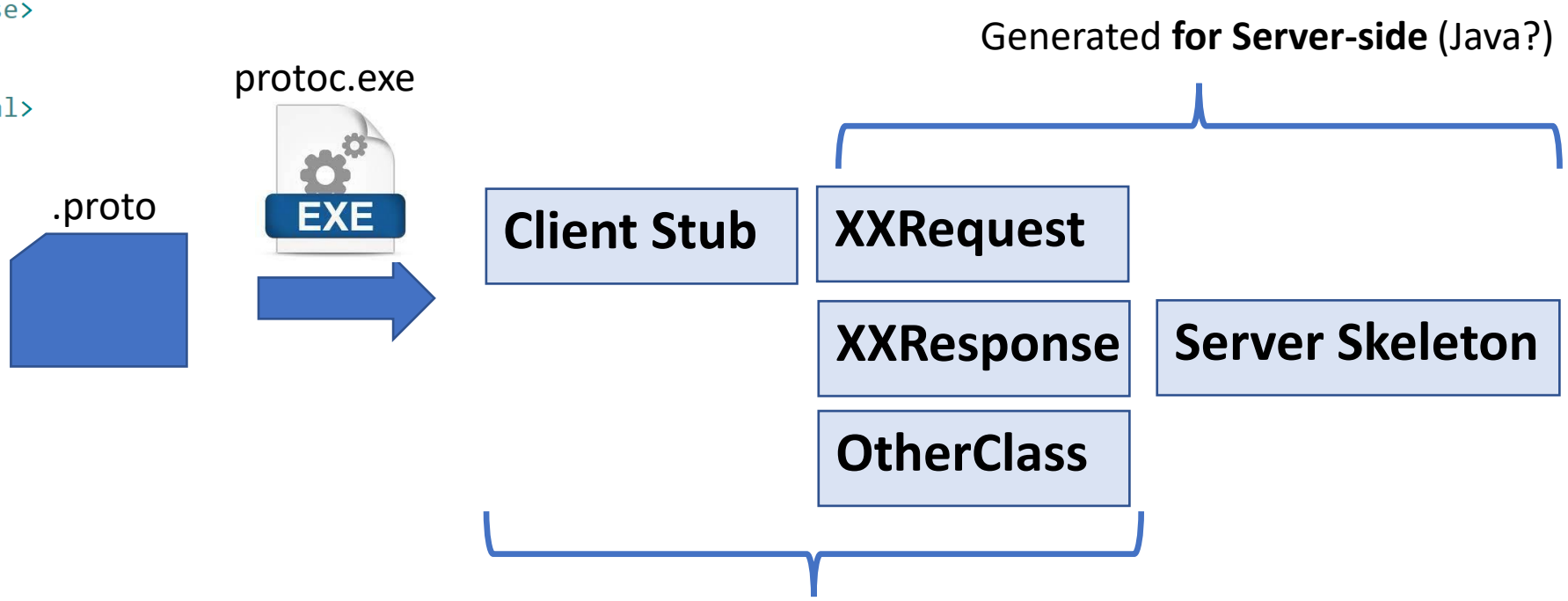
<?xml>	⬅	xsd schema
{ json}	⬅	?? No standard (json-schema,OpenApi,Swagger)
CSV	⬅	Header line … Tabular, Not tree

Binary
+ code generator
per-langage

Corba,RPC,XRD,..	⬅	.idl
Avro	⬅	.avsc : Avro-Schema (in json)
Thrift	⬅	.idl : Thrift Schema
Protobuf,gRPC	⬅	.proto : Protobuf Schema

Example Code-Generator: gPRC (protobuf-maven-plugin)

```xml
<plugin>
    <groupId>org.xolstice.maven.plugins</groupId>
    <artifactId>protobuf-maven-plugin</artifactId>
    <version>0.6.1</version>
    <configuration>
        <protocArtifact>com.google.protobuf:protoc:${protoc.version}:exe:${os.detected.classifier}</protocArtifact>
        <pluginId>grpc-java</pluginId>
        <pluginArtifact>io.grpc:protoc-gen-grpc-java:${grpc.version}:exe:${os.detected.classifier}</pluginArtifact>
    </configuration>
    <executions>
        <execution>
            <phase>generate-sources</phase>
            <goals>
                <goal>compile</goal>
                <goal>compile-custom</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```



Generated Code: POJO + Builder + writeTo(Out..) / parseFrom(In..)

```
message HelloMessage {

  string  fieldString1  = 1;

  int32   fieldInt2     = 2;
  int64   fieldLong3    = 3;

  float   fieldFloat4   = 4;
  double  fieldDouble5  = 5;

  bool    fieldBool6    = 6;

}
```

protoc.exe

- FIELDSTRING1_FIELD_NUMBER : int
- fieldString1_ : Object
- getFieldString1() : String
- getFieldString1Bytes() : ByteString
- FIELDINT2_FIELD_NUMBER : int
- fieldInt2_ : int
- getFieldInt2() : int
- FIELDLONG3_FIELD_NUMBER : int
- fieldLong3_ : long
- getFieldLong3() : long
- FIELDFLOAT4_FIELD_NUMBER : int
- fieldFloat4_ : float
- getFieldFloat4() : float
- FIELDDOUBLE5_FIELD_NUMBER : int
- fieldDouble5_ : double
- getFieldDouble5() : double
- FIELDBOOL6_FIELD_NUMBER : int
- fieldBool6_ : boolean
- getFieldBool6() : boolean

- newBuilderForType() : Builder
- newBuilder() : Builder
- newBuilder(HelloMessage) : Builder
- toBuilder() : Builder
- Builder
  - getDescriptor() : Descriptor
  - clear() : Builder
  - getDescriptorForType() : Descriptor
  - getDefaultInstanceForType() : HelloMessage
  - build() : HelloMessage
  - buildPartial() : HelloMessage
  - clone() : Builder
  - setField(FieldDescriptor, Object) : Builder

- writeTo(CodedOutputStream) : void
- getSerializedSize() : int

- parseFrom(ByteBuffer) : HelloMessage
- parseFrom(ByteBuffer, ExtensionRegistryLite) : HelloMessage
- parseFrom(ByteString) : HelloMessage
- parseFrom(ByteString, ExtensionRegistryLite) : HelloMessage
- parseFrom(byte[]) : HelloMessage
- parseFrom(byte[], ExtensionRegistryLite) : HelloMessage
- parseFrom(InputStream) : HelloMessage
- parseFrom(InputStream, ExtensionRegistryLite) : HelloMessa
- parseDelimitedFrom(InputStream) : HelloMessage
- parseDelimitedFrom(InputStream, ExtensionRegistryLite) : He
- parseFrom(CodedInputStream) : HelloMessage
- parseFrom(CodedInputStream, ExtensionRegistryLite) : Hello

writeTo() = efficient binary Marshalling
parseFrom() = .. Un-Marshalling

```
@java.lang.Override
public void writeTo(com.google.protobuf.CodedOutputStream output)
                    throws java.io.IOException {
  if (!getFieldString1Bytes().isEmpty()) {
    com.google.protobuf.GeneratedMessageV3.writeString(output, 1, fieldString1_);
  }
  if (fieldInt2_ != 0) {
    output.writeInt32(2, fieldInt2_);
  }
  if (fieldLong3_ != 0L) {
    output.writeInt64(3, fieldLong3_);
  }
  if (fieldFloat4_ != 0F) {
    output.writeFloat(4, fieldFloat4_);
  }
  if (fieldDouble5_ != 0D) {
    output.writeDouble(5, fieldDouble5_);
  }
  if (fieldBool6_ != false) {
    output.writeBool(6, fieldBool6_);
  }
  unknownFields.writeTo(output);
}
```

Sample Server Implementation

```java
import fr.an.tests.testpgrpc.GreeterGrpc;
import fr.an.tests.testpgrpc.HelloReply;
import fr.an.tests.testpgrpc.HelloRequest;

import io.grpc.stub.StreamObserver;

public final class GrpcImplService extends GreeterGrpc.GreeterImplBase {

    @Override
    public void sayHello(HelloRequest req, StreamObserver<HelloReply> responseObserver) {
        String reqName = req.getName();
        log.info("sayHello req.name:" + reqName);
        HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + reqName + ", from " + serverName).build();

        log.info(".. response (onNext+onComplete) sayHello req.name:" + reqName);
        responseObserver.onNext(reply);
        responseObserver.onCompleted();
    }

    @Override
    public void sayHelloOutStream(HelloRequest req, StreamObserver<HelloReply> responseObserver) {
        String reqName = req.getName();
        log.info("sayHello req.name:" + reqName);
        HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + reqName + ", from " + serverName).build();

        log.info(".. response 5x onNext + 5x (sleep + onNext) + onComplete) sayHello req.name:" + reqName);
        for(int i = 0; i < 5; i++) {
            responseObserver.onNext(reply);
        }
        for(int i = 0; i < 5; i++) {
            sleep(1000);
            responseObserver.onNext(reply);
        }

        responseObserver.onCompleted();
    }
```

Sample Client Calls

```
    Channel channel = ManagedChannelBuilder.forAddress(host, port).usePlaintext().build();
    this.blockingStub = GreeterGrpc.newBlockingStub(channel);
    this.asyncStub = GreeterGrpc.newFutureStub(channel);
    this.stub = GreeterGrpc.newStub(channel);
    Log.info("call to channel.. " + channel);
}


public void sayHello_blockingStub() throws Exception {
    HelloRequest helloReq = HelloRequest.newBuilder().setName("you").build();

    HelloReply reply = blockingStub.sayHello(helloReq);

    Log.info("sayHello => " + reply.getMessage());
}
```

gRPC … Async for performance

(see also java concurrent.Future / ReactiveJava / ProjectReactor …)

Details

```
package io.grpc.stub;

 * Receives notifications from an observable stream of messages.
public interface StreamObserver<V>  {
   * Receives a value from the stream.
  void onNext(V value);

   * Receives a terminating error from the stream.
  void onError(Throwable t);

   * Receives a notification of successful stream completion.
  void onCompleted();
}
```

Sample Client Calls... Async

Details

```
public void sayHello_asyncStub_get() throws Exception {
    HelloRequest helloReq = HelloRequest.newBuilder().setName("you").build();
    ListenableFuture<HelloReply> respListenableFuture = asyncStub.sayHello(helloReq);

    // wrap in java.util.CompletableFuture, use .thenApply(), thenCombine(), ...
    CompletableFuture<HelloReply> completableFuture = MoreFutures.toCompletableFuture(respListenableFuture);
    CompletableFuture<HelloReply> resFuture = completableFuture// .thenCombine(..) // combine with other futures
        .thenApply(x -> x); // transform when ready

    HelloReply reply = resFuture.get(); // (generally don't do that) async to blocking!
    log.info("sayHello (async block) => " + reply.getMessage());
}

public void sayHello_stub() throws Exception {
    HelloRequest helloReq = HelloRequest.newBuilder().setName("you").build();
    CountDownLatch latch = new CountDownLatch(1);
    stub.sayHello(helloReq, new StreamObserver<HelloReply>() {
        @Override
        public void onNext(HelloReply reply) {
            log.info("sayHello .. response => " + reply.getMessage());
        }
        @Override
        public void onCompleted() {
            log.info("sayHello .. onComplete");
            latch.countDown();
        }
        @Override
        public void onError(Throwable t) {
            log.info("sayHello .. onError", t);
            latch.countDown();
        }
    });
    latch.await();
    log.info(".. done wait sayHello");
}
```

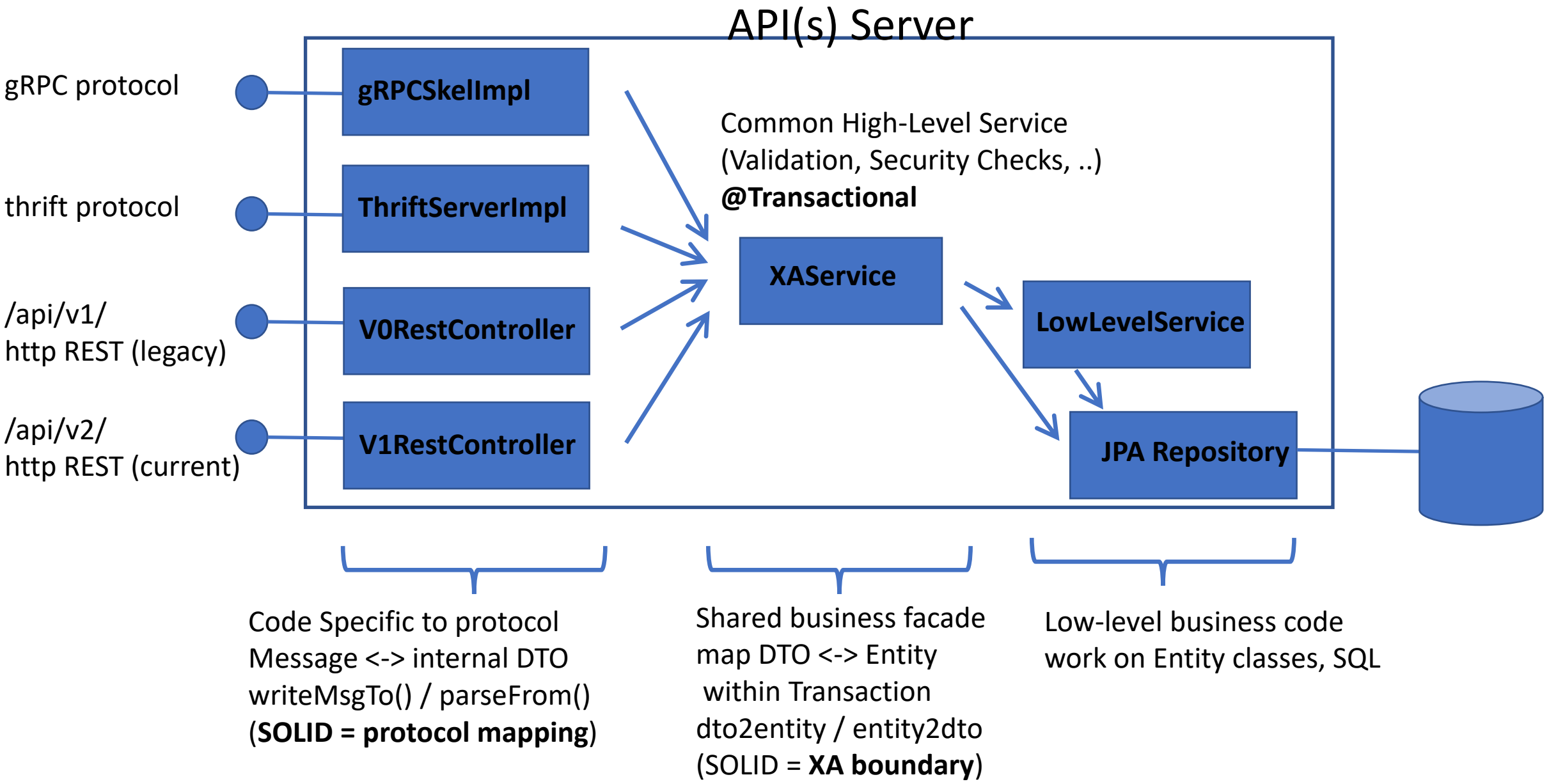
Generalizing gRPC « pattern skeleton code » (call Service … Entity to DTO … to Message)

```
@Autowired
FooService delegateXAService;

@Override
public void sayHello(HelloRequest req, StreamObserver<HelloReply> responseObserver) {
    // Step 1/3:
    // validity check, extract/convert gRPC Request to internal classes (internal DTO)
    long startTime = System.currentTimeMillis();
    String reqName = req.getName();
    log.info("sayHello name:" + reqName);
    FooDTO inDto = new FooDTO(reqName);

    // Step 2/3:
    // delegate to transactional service  (SOLID principle)
    FooDTO tmpres = delegateXAService.saveHello(inDto);

    // Step 3/3:
    // convert internal DTO to gRPC response, marshall response
    long millis = System.currentTimeMillis() - startTime;
    log.info(".. done sayHello, took " + millis);
    HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + tmpres.message).build();
    responseObserver.onNext(reply);
    responseObserver.onCompleted();
}
```

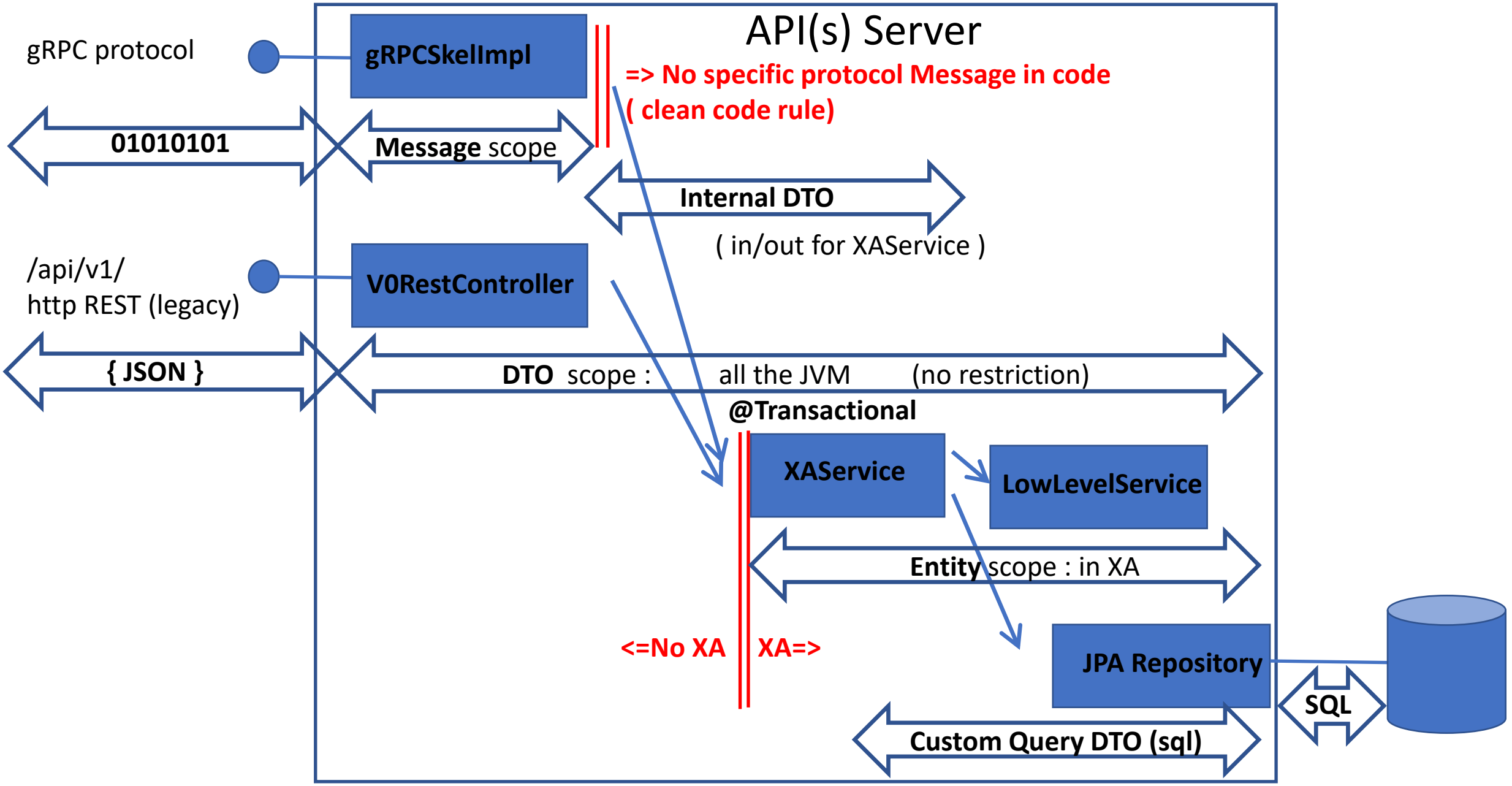
Generalizing Server-side « pattern skeleton code »

```
@XXXProtocolMapping(path="/sayHello")
public void sayHello(
        @XXXRequestBody HelloRequest req,
        @XXXResponse XXXOutputResponse out) {
    // Step 1/3:
    // validity check, extract/convert XXX Request to internal classes (internal DTO)
    long startTime = System.currentTimeMillis();
    String reqName = req.getName();
    Log.info("sayHello name:" + reqName);
    FooDTO inDto = new FooDTO(reqName);

    // Step 2/3:
    // delegate to transactional service  (SOLID principle)
    FooDTO tmpres = delegateXAService.saveHello(inDto);

    // Step 3/3:
    // convert internal DTO to XXX response, marshall response
    long millis = System.currentTimeMillis() - startTime;
    Log.info(".. done sayHello, took " + millis);
    HelloReply reply = HelloReply.newBuilder().setMessage("Hello " + tmpres.message).build();
    out.status(200).addHeader("header1", "value1").body(reply);
}
```

Design code for several protocols / versionned APIs



Message / Api DTO != Internal DTO



gRPC / protobuf
… designed for backward compatibility

Zooming on backward compatibility
… ideas for Rest / Json backward compatibility ?

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 fields are numbered (unique ID)

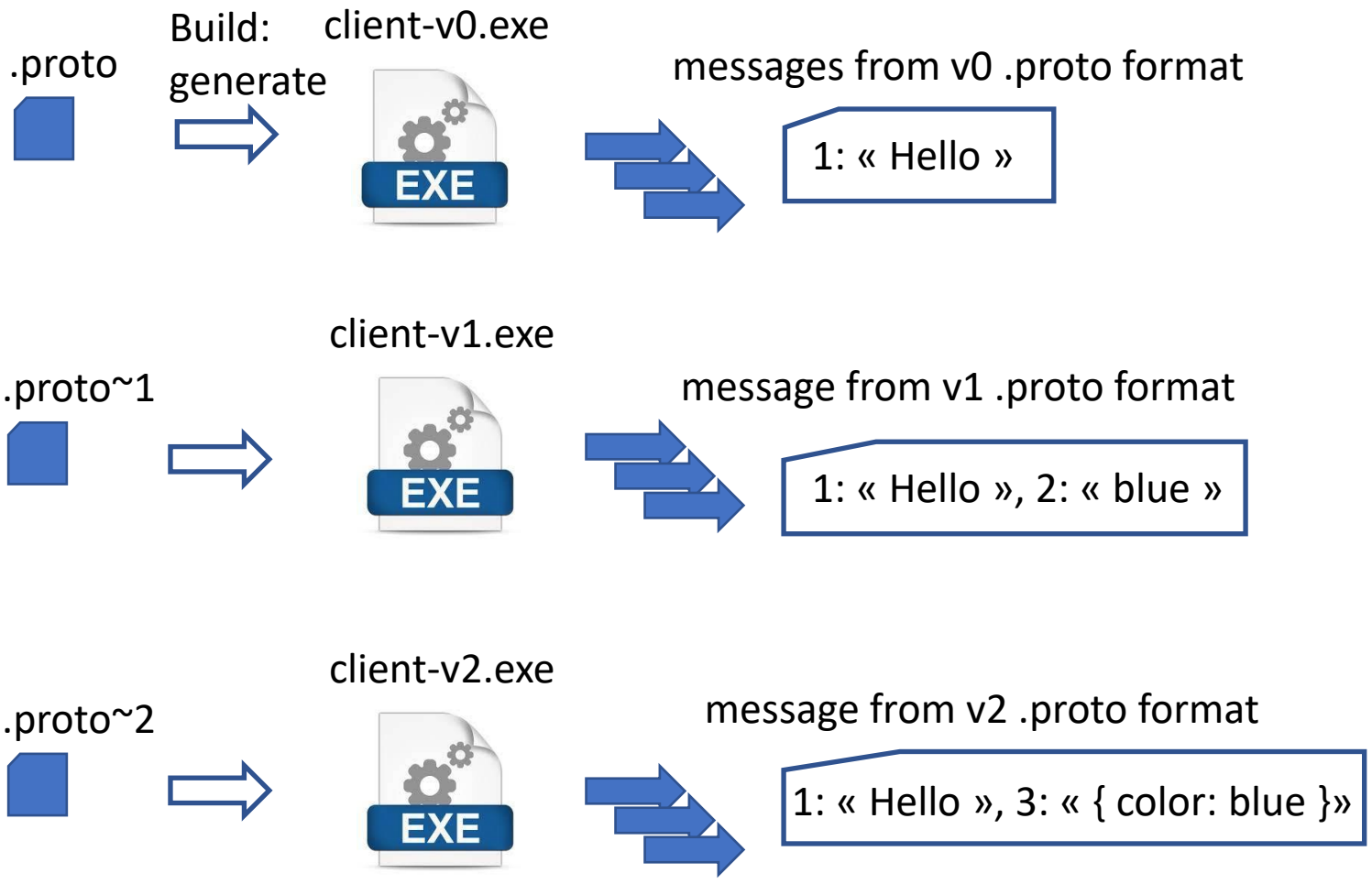
```
message HelloRequest {
  string name = 1;
}


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

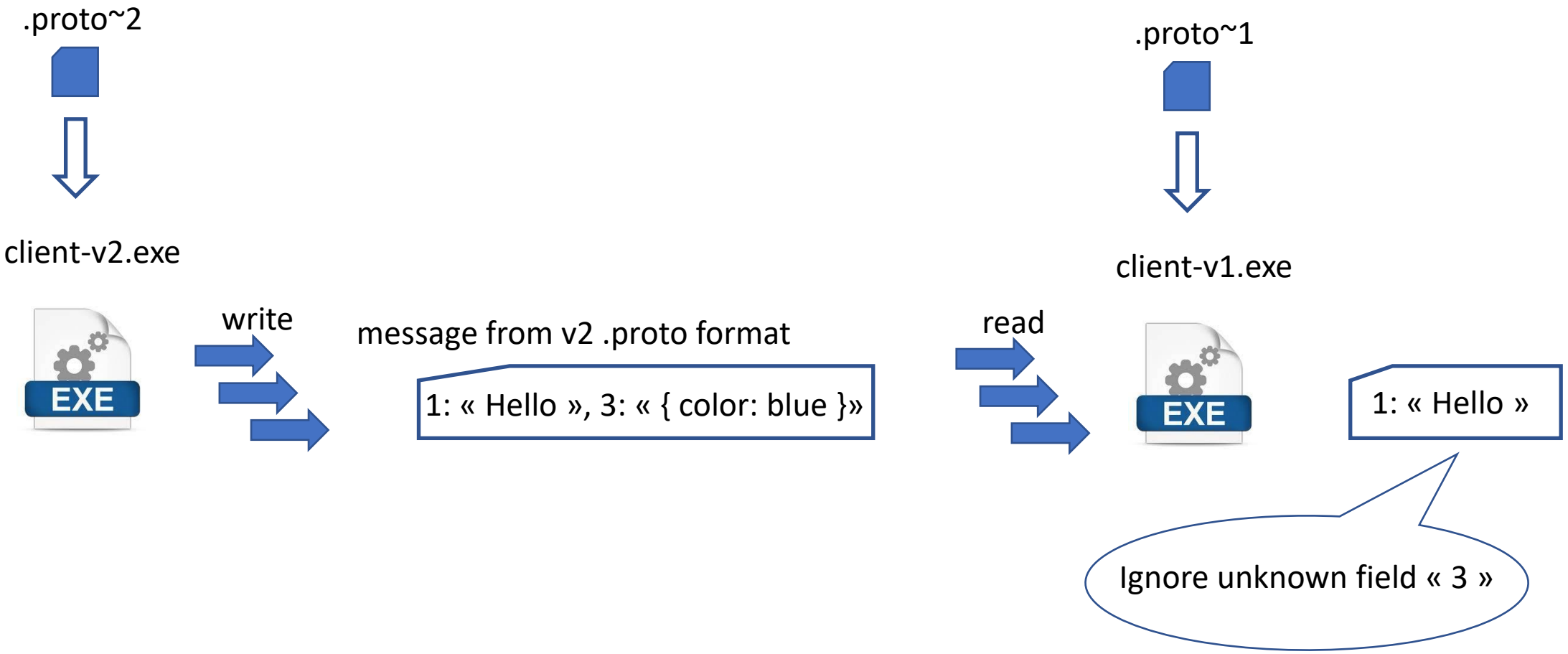
In Messages
« Map<FieldId, Value> »

fieldId looks « unneccessary »

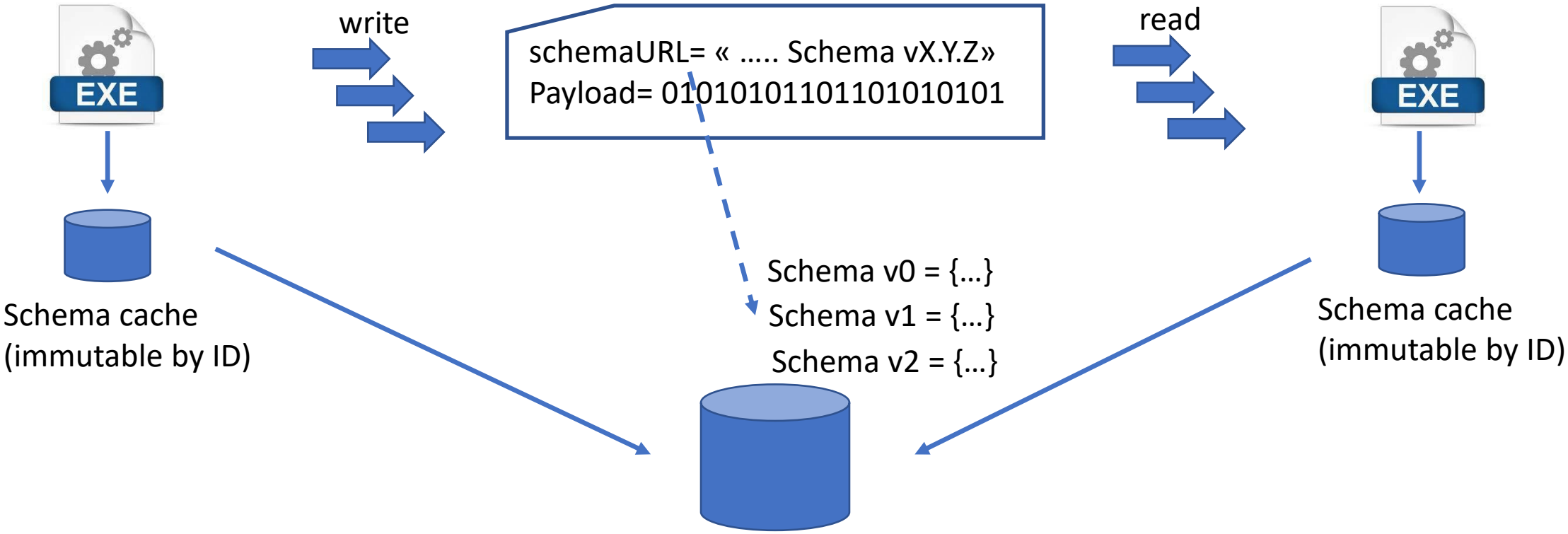
Small extra cost
… Great compatibility



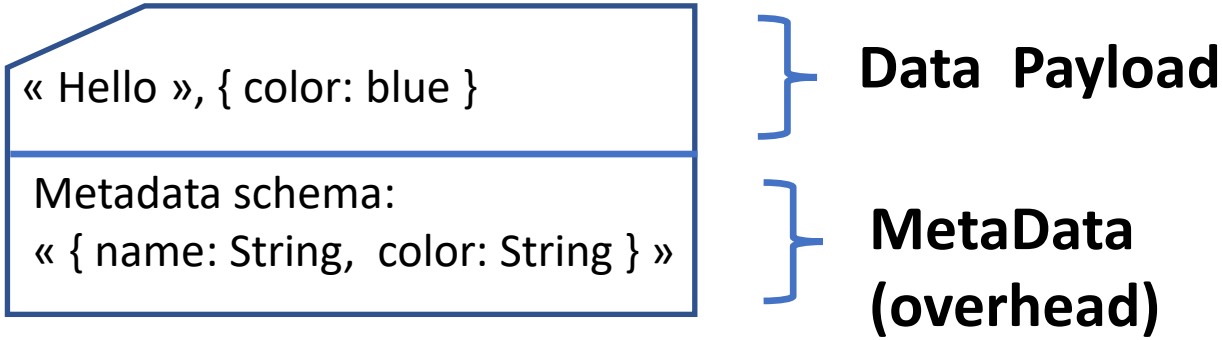
Reading « Future » Message / Backward Client … Ignore Unknown Fields



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)

Binary (protobuf,..) with Schema = OK

What about JSON ?
Schema-Less / No code generator

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

Script (untyped interpreter)

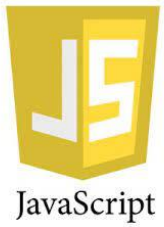
```
let object = {
    name: 'arnaud',
    skills: [ 'IT', 'math']
};
console.log(`Hello ${object.name}`);

// 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;
}
```



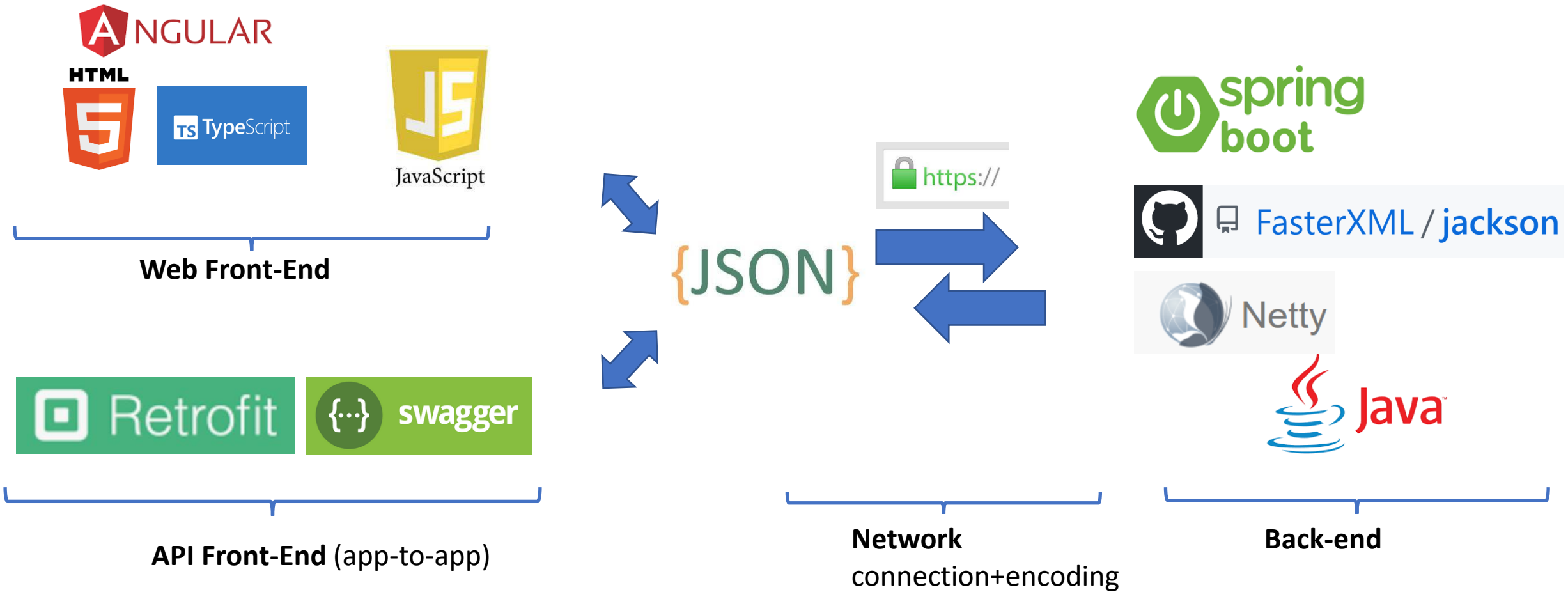
{JSON}

Web Front-End

Network encoding

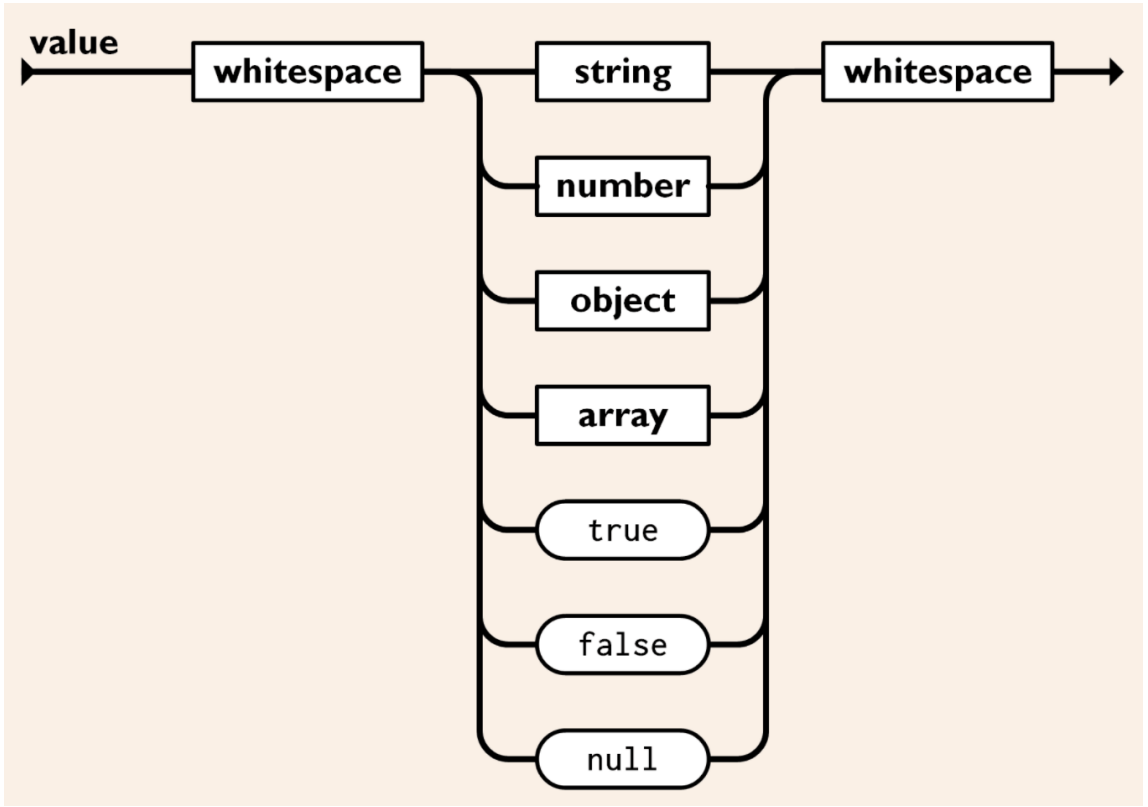
Back-end

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

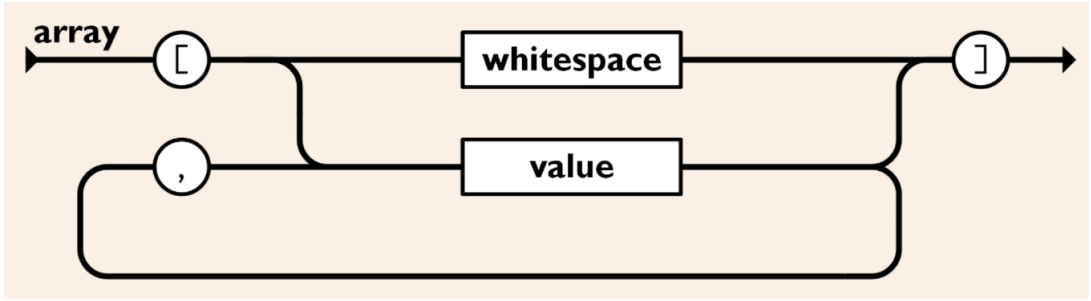


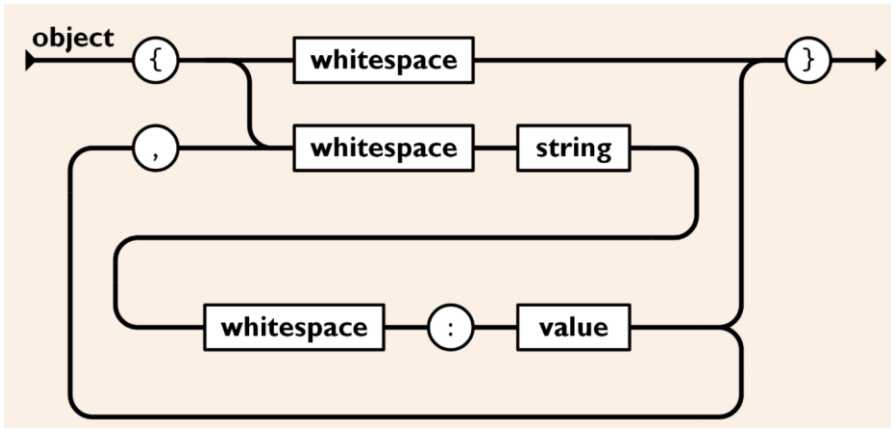
Grammar of JSON
https://www.json.org/

Literal value (terminal)

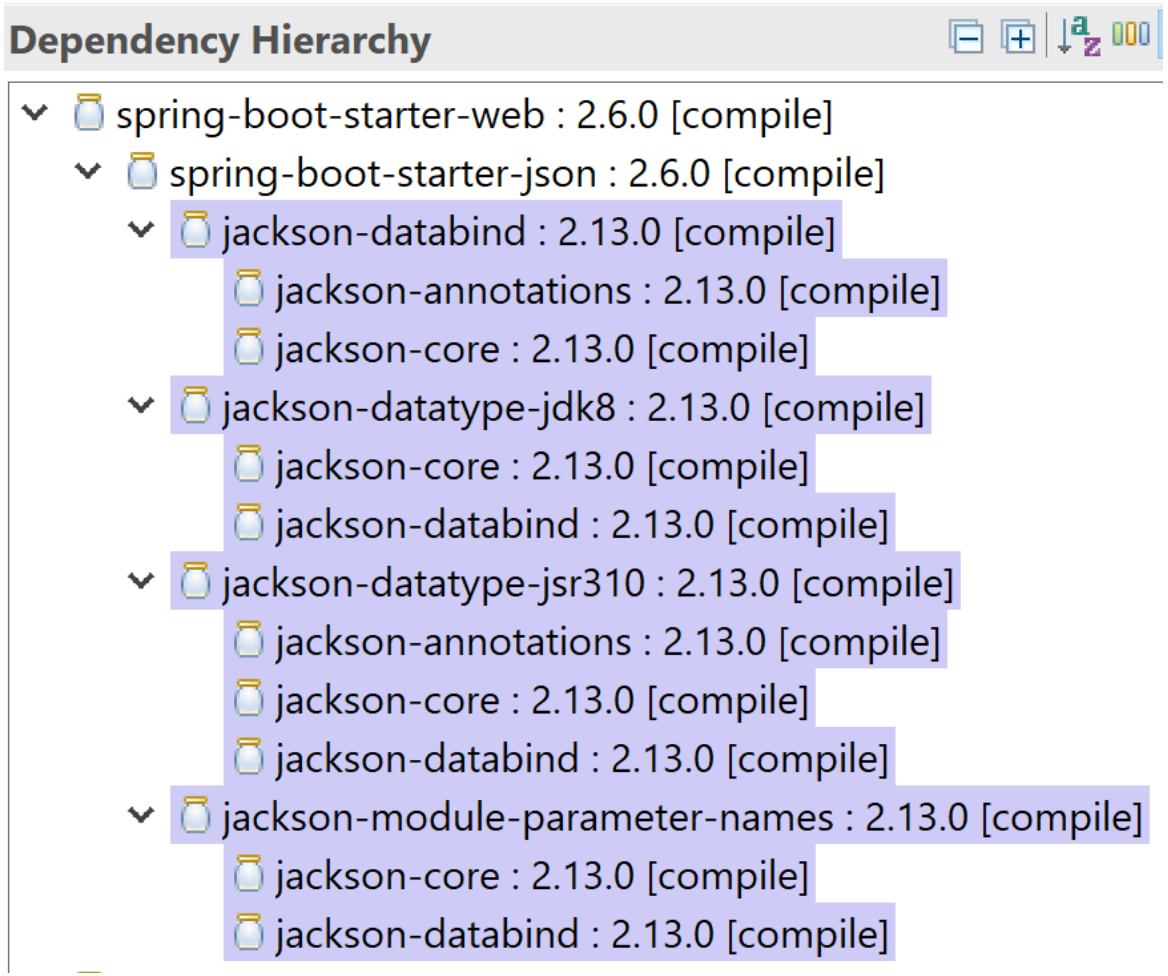


Array / Object composition





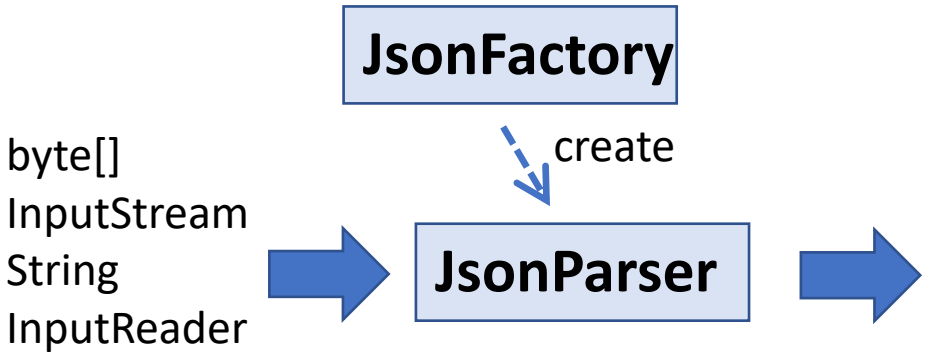
Jackson
(default dependencies from spring)



Jackson-Core: Json Streaming api for Parsing

jackson-core-2.13.0.jar

com.fasterxml.jackson.core



JsonToken
- NOT_AVAILABLE
- START_OBJECT
- END_OBJECT
- START_ARRAY
- END_ARRAY
- FIELD_NAME
- VALUE_EMBEDDED_OBJECT
- VALUE_STRING
- VALUE_NUMBER_INT
- VALUE_NUMBER_FLOAT
- VALUE_TRUE
- VALUE_FALSE
- VALUE_NULL

- nextToken() : JsonToken
- currentToken() : JsonToken
- currentTokenId() : int

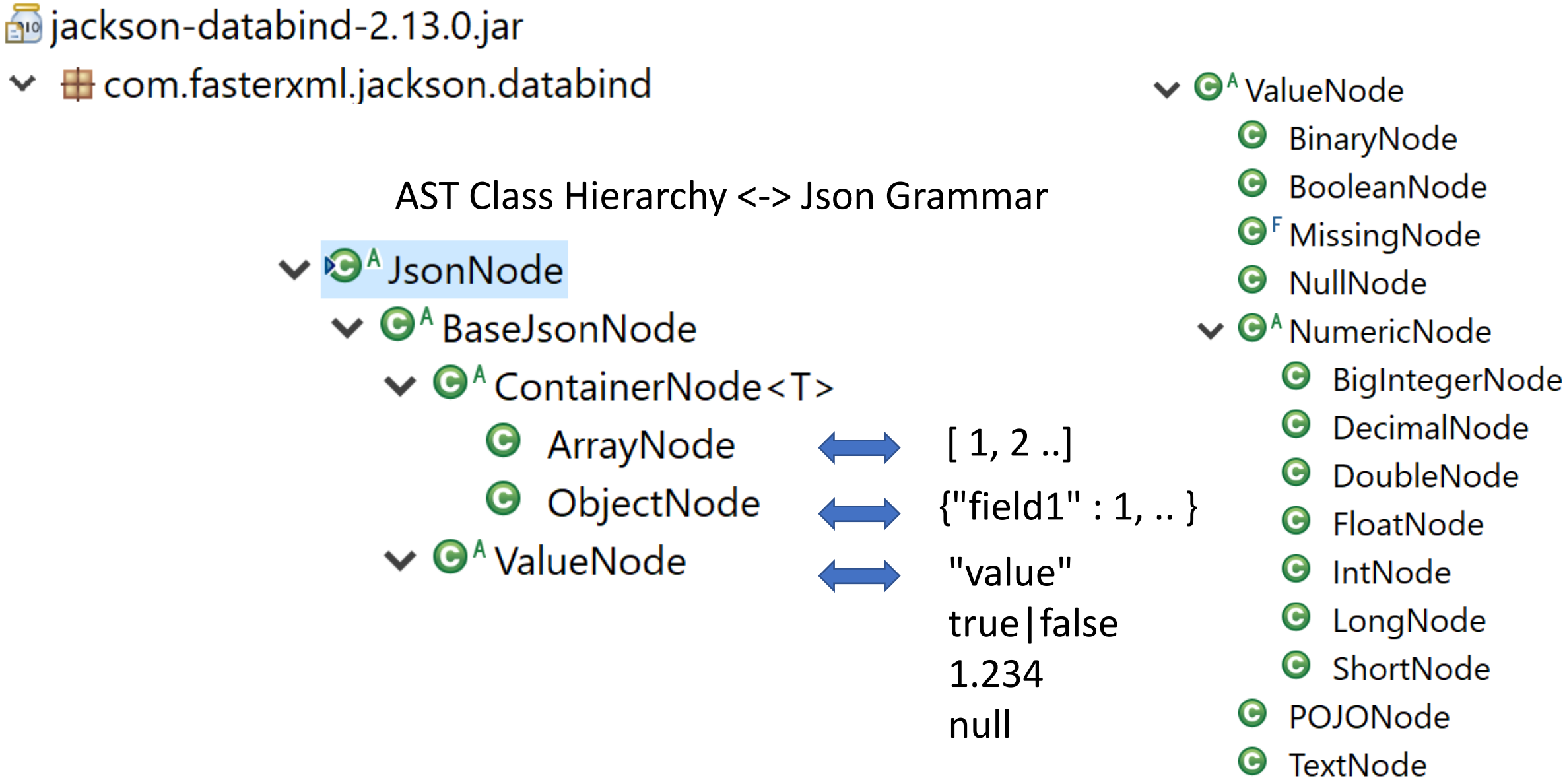
- getValueAsInt() : int
- getValueAsLong() : long
- getValueAsString() : String

..

Sample Json Streaming Parsing

```
String json = "[ 123, true, { \"f\": \"abcd\" } ]";

JsonFactory jsonFactory = new JsonFactory();
JsonParser parser = jsonFactory.createParser(json);

JsonToken tk;
tk = parser.nextToken(); assertEquals(JsonToken.START_ARRAY, tk);
tk = parser.nextToken(); assertEquals(JsonToken.VALUE_NUMBER_INT, tk);
int i = parser.getIntValue(); assertEquals(123, i);

tk = parser.nextToken(); assertEquals(JsonToken.VALUE_TRUE, tk);

tk = parser.nextToken(); assertEquals(JsonToken.START_OBJECT, tk);
tk = parser.nextToken(); assertEquals(JsonToken.FIELD_NAME, tk);
String name = parser.getCurrentName(); assertEquals("f", name);
tk = parser.nextToken(); assertEquals(JsonToken.VALUE_STRING, tk);
String text = parser.getText(); assertEquals("abcd", text);
tk = parser.nextToken(); assertEquals(JsonToken.END_OBJECT, tk);

tk = parser.nextToken(); assertEquals(JsonToken.END_ARRAY, tk);
tk = parser.nextToken(); assertNull(tk);
```

Jackson-databind : JsonNode Class Hierarchy



Parsing Json to in-memory Tree (no user-defined class)

{ JSON} → JsonNode Tree

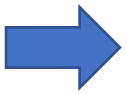
JsonNode Tree → { JSON}

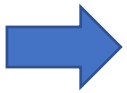
Could not use typed List<T>
.. only List<JsonNode>

```
String json = "[123, { \"name\": \"abcd\" }]";

ObjectMapper om = new ObjectMapper();
JsonNode tree = om.readTree(json);

assertEquals(JsonToken.START_ARRAY, tree.asToken());
JsonNode elt0 = tree.get(0), elt1 = tree.get(1);
assertEquals(123, elt0.asInt());
assertEquals(JsonToken.START_OBJECT, elt1.asToken());
JsonNode nameNode = elt1.findValue("name");
assertEquals(JsonToken.VALUE_STRING, nameNode.asToken());
assertEquals("abcd", nameNode.asText());
```

```
JsonNodeFactory f = new JsonNodeFactory(true);
ArrayNode arrayNode = f.arrayNode();
arrayNode.add(f.numberNode(123));
ObjectNode objectNode = f.objectNode();
objectNode.set("name", f.textNode("abcd"));
arrayNode.add(objectNode);

String json = arrayNode.toString();
// arrayNode.toPrettyString();

assertEquals("[123,{\"name\":\"abcd\"}]", json);
```

Jackson-databind: ObjectMapper

{ JSON}  Object

Object  { JSON}

```
public class User {
    public String name;
}
```

```
String json = "{ \"name\": \"abcd\" }";

ObjectMapper om = new ObjectMapper();
User bean = om.readValue(json, User.class);

assertEquals("abcd", bean.name);
```

```
User bean = new User();
bean.name = "abcd";

ObjectMapper om = new ObjectMapper();
String json = om.writeValueAsString(bean);

assertEquals("{\"name\":\"abcd\"}", json);
```

Type discriminant for « class / sub-classes »

```java
@JsonTypeInfo(use=Id.NAME, property="type")
@JsonSubTypes({
    @Type(name="dog", value=Dog.class),
    @Type(name="duck", value=Duck.class),
})
public static abstract class Animal {

}
```

JSON

```
{"type":"dog", "barking":"houah"}
```

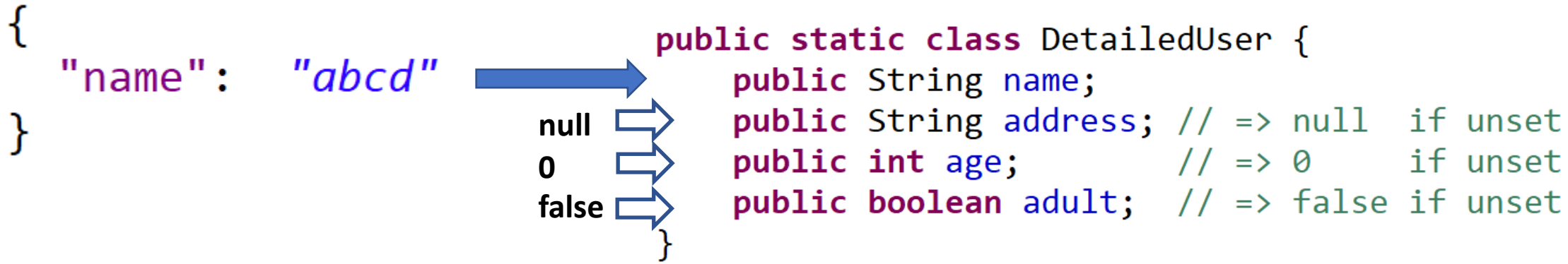
```java
public static class Dog extends Animal {
    public String barking;
}
```

```
{"type":"duck", "quacking":"coin"}
```

```java
public static class Duck extends Animal {
    public String quacking;
```

```java
Dog dog = new Dog();
dog.barking = "houah";
Animal animal = dog;

ObjectMapper om = new ObjectMapper();
String json = om.writeValueAsString(animal);

assertEquals("{\"type\":\"dog\",\"barking\":\"houah\"}", json);

Animal a = om.readValue(json, Animal.class); // parse any from abstract class
assertTrue(a instanceof Dog); // instanciated sub-class
assertEquals("houah", ((Dog) a).barking);
```

Field not set (not in JSON) => 0, false, null (in Java)



disable FAIL Unknown properties

```
String json = "{ \"name\": \"abcd\", \"xxx\": 123 }";

ObjectMapper om = new ObjectMapper();
om.disable(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES); // override
User bean = om.readValue(json, User.class);

assertEquals("abcd", bean.name);
// field "xxx" not mapped to bean
```

```
String json = "{ \"name\": \"abcd\", \"xxx\": 123 }";

ObjectMapper om = new ObjectMapper();
// om.enable(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES); // default
try {
    om.readValue(json, User.class);
    Assert.fail(); //
} catch(UnrecognizedPropertyException ex) {
    // ok !
}
```

Ignored / Unset / Unknown fields

```
{
  "field1":  "value1",
  "field2":  "value2",
```

```
class UserDefinedDTO {
  public String field1;
  public String field2;
```

(not in Json) UNSET… 0

```
  public String field5;
  public String field6;
```

```
  "unkownField3": "value3",
  "unkownField4": "value4"
}
```

UNKOWN (not in Java)

```
  // private, no getter => ignored
  private String ignoreField7;
```

Ignored in java

```
  @JsonIgnore // explicitly ignored
  public String ignoreField8;
}
```

Mixing Generic JsonNode in Type-safe DTO

{ JSON} → DTO with JsonNode

DTO with JsonNode → { JSON}

```java
public class UntypedDataDTO {
    public String field1;
    public JsonNode untypedData;
}
```

```java
String json = "{\"field1\":\"abcd\",\"untypedData\":[123,\"abc\"]}";

ObjectMapper om = new ObjectMapper();
UntypedDataDTO bean = om.readValue(json, UntypedDataDTO.class);

assertEquals("abcd", bean.field1);
ArrayNode arrayNode = (ArrayNode) bean.untypedData;
assertEquals(2, arrayNode.size());
JsonNode elt0 = arrayNode.get(0), elt1 = arrayNode.get(1);
assertEquals(123, elt0.asInt());
assertEquals("abc", elt1.asText());
```

```java
UntypedDataDTO bean = new UntypedDataDTO();
bean.field1 = "abcd";
JsonNodeFactory f = new JsonNodeFactory(true);
ArrayNode arrayNode = f.arrayNode();
arrayNode.add(f.numberNode(123));
arrayNode.add(f.textNode("abc"));
bean.untypedData = arrayNode;

ObjectMapper om = new ObjectMapper();
String json = om.writeValueAsString(bean);

assertEquals("{\"field1\":\"abcd\",\"untypedData\":[123,\"abc\"]}", json
```

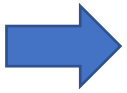
Support for Any Getter/Setter Properties

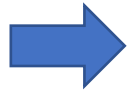
Known properties GO here

All **others** unknown properties GO here

```
public class ExtraDataDTO {
    public String field1;

    private Map<String,Object> extraData = new LinkedHashMap<>();
    public Object getExtraData(String key) {
        return extraData.get(key);
    }

    @JsonAnyGetter
    public Map<String,Object> getExtraData() {
        return extraData;
    }
    @JsonAnySetter
    public void setExtraData(String key, Object value) {
        this.extraData.put(key, value);
    }
}
```

Any Getter/Setter Properties … preserve future Unknown

{ JSON}  DTO with Any Property

DTO with Any Property  { JSON}

```
String json = "{" +
        "\"field1\":\"abcd\"," +
        "\"field2\":\"bcde\"," +
        "\"field3\":123" +
        "}";

ObjectMapper om = new ObjectMapper();
ExtraDataDTO bean = om.readValue(json, ExtraDataDTO.class);

assertEquals("abcd", bean.field1);
Object value2 = bean.getExtraData("field2");
Object value3 = bean.getExtraData("field3");
assertEquals(2, bean.getExtraData().size());
assertEquals(123, ((Integer) value2).intValue());
assertEquals("abc", (String) value3);
```

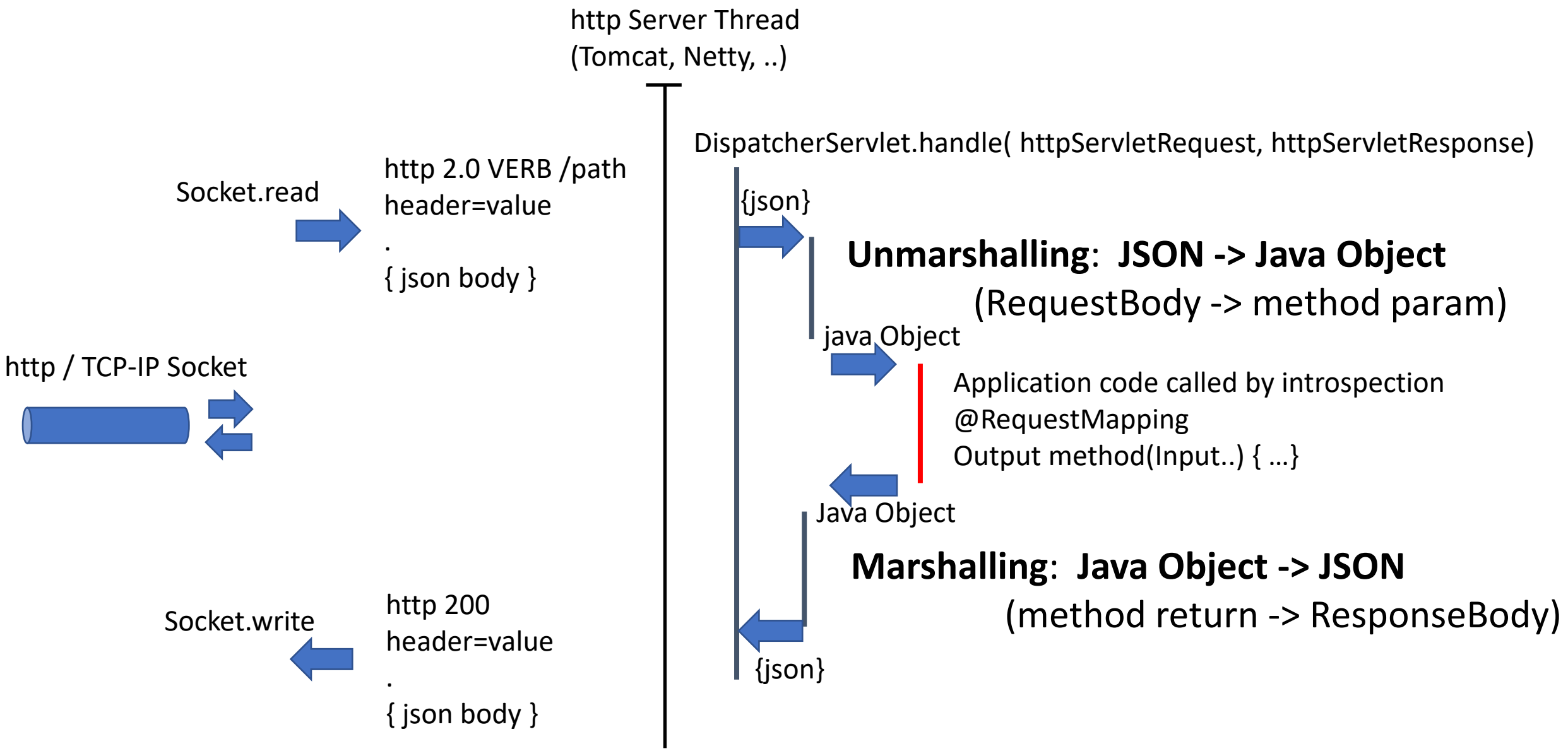
```
ExtraDataDTO bean = new ExtraDataDTO();
bean.field1 = "abcd";
bean.setExtraData("field2", "bcde");
bean.setExtraData("field3", 123);

ObjectMapper om = new ObjectMapper();
String json = om.writeValueAsString(bean);

String expectedJson = "{" +
        "\"field1\":\"abcd\"," +
        "\"field2\":\"bcde\"," +
        "\"field3\":123" +
        "}";
assertEquals(expectedJson, json);
```

JSON : schema-less
well integrated in Java (Jackson)

even more portable than protobuf
but much more verbose…

Mapping Rest Request <-> server Method Call

Http Json Request <-> Java method + Object param

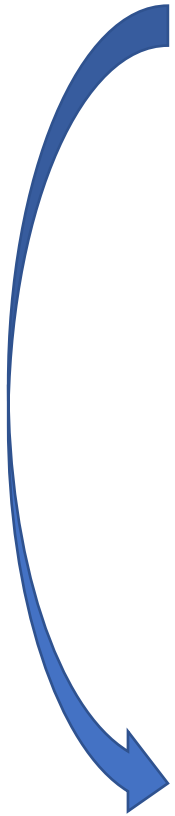


Http Request <-> Spring Java Mappings
@RequestMapping, @{Get|Post|..}Mapping, @RequestBody ..

```java
@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;
}
```

```java
@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



```java
@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

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

```java
@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

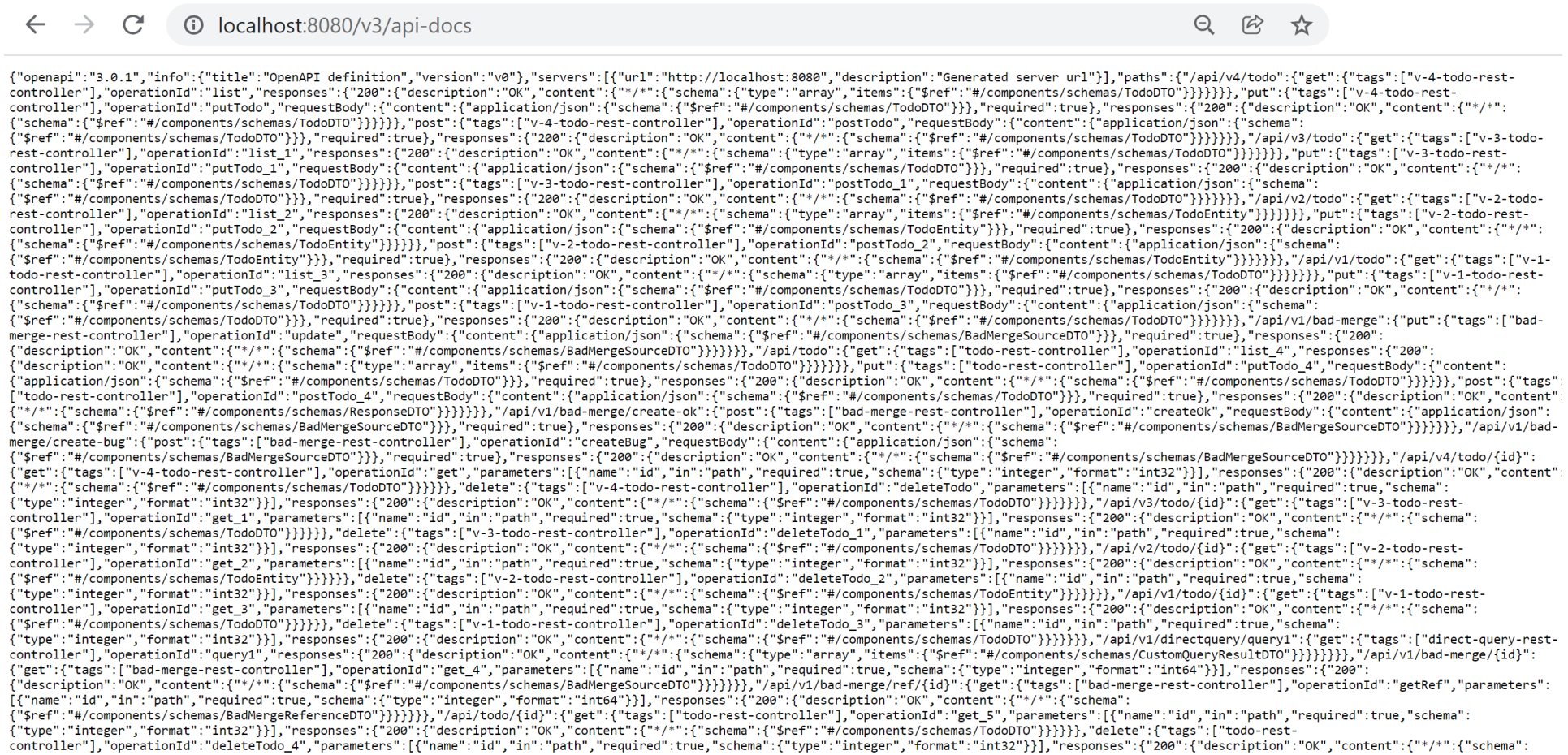
```
// implicit equivalent.. with extra header
@GetMapping(path = "/equivalent2/{id}",
        produces = "application/json")
public ResponseEntity<byte[]> get2(@PathVariable("id") int id) throws JsonProcessingException {
    TodoDTO res = service.get(id);
    byte[] content = jsonMapper.writeValueAsBytes(res);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

Documenting Rest APIs
Exposing Rest Api « Metadata »

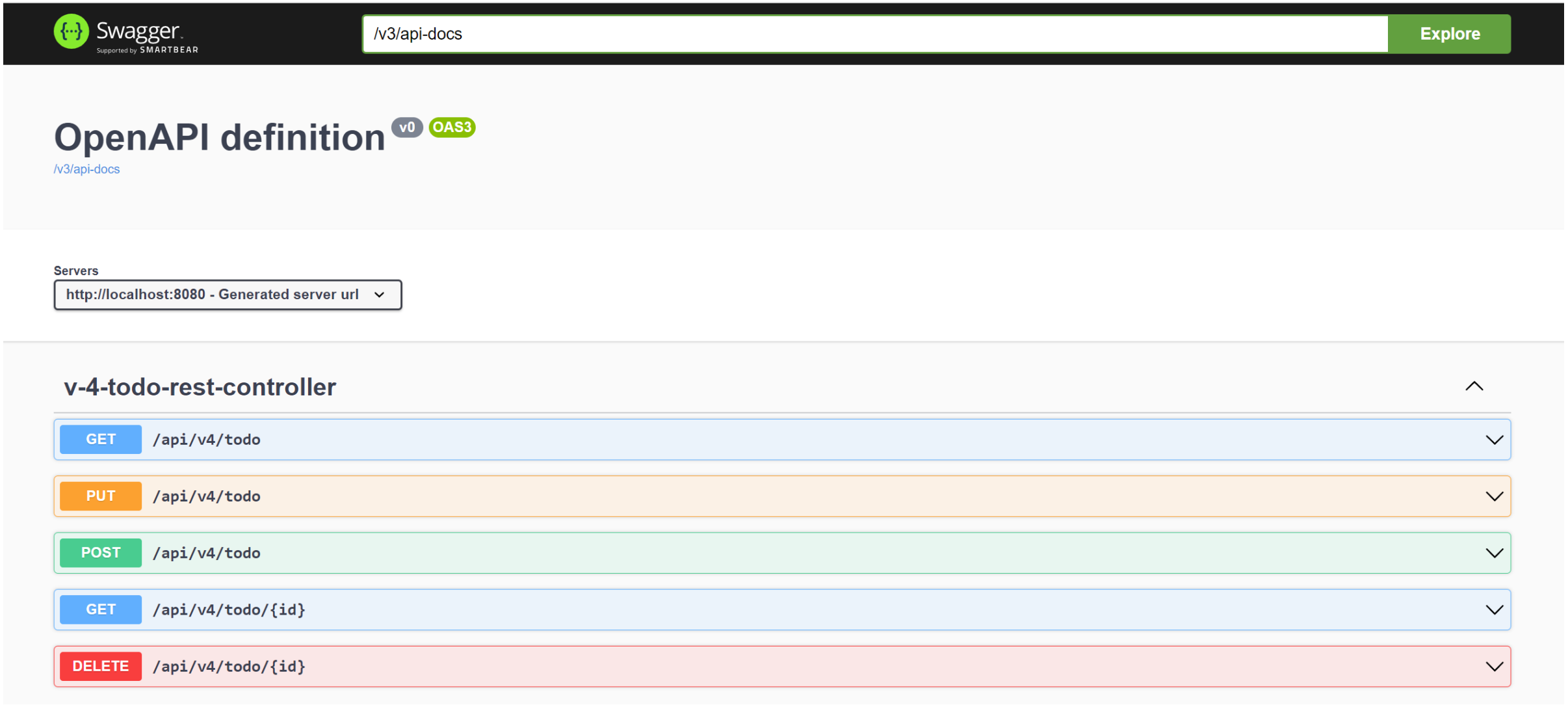
Springdoc-openapi-ui

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-ui</artifactId>
    <version>${springdoc-openapi-ui.version}</version>
</dependency>
```

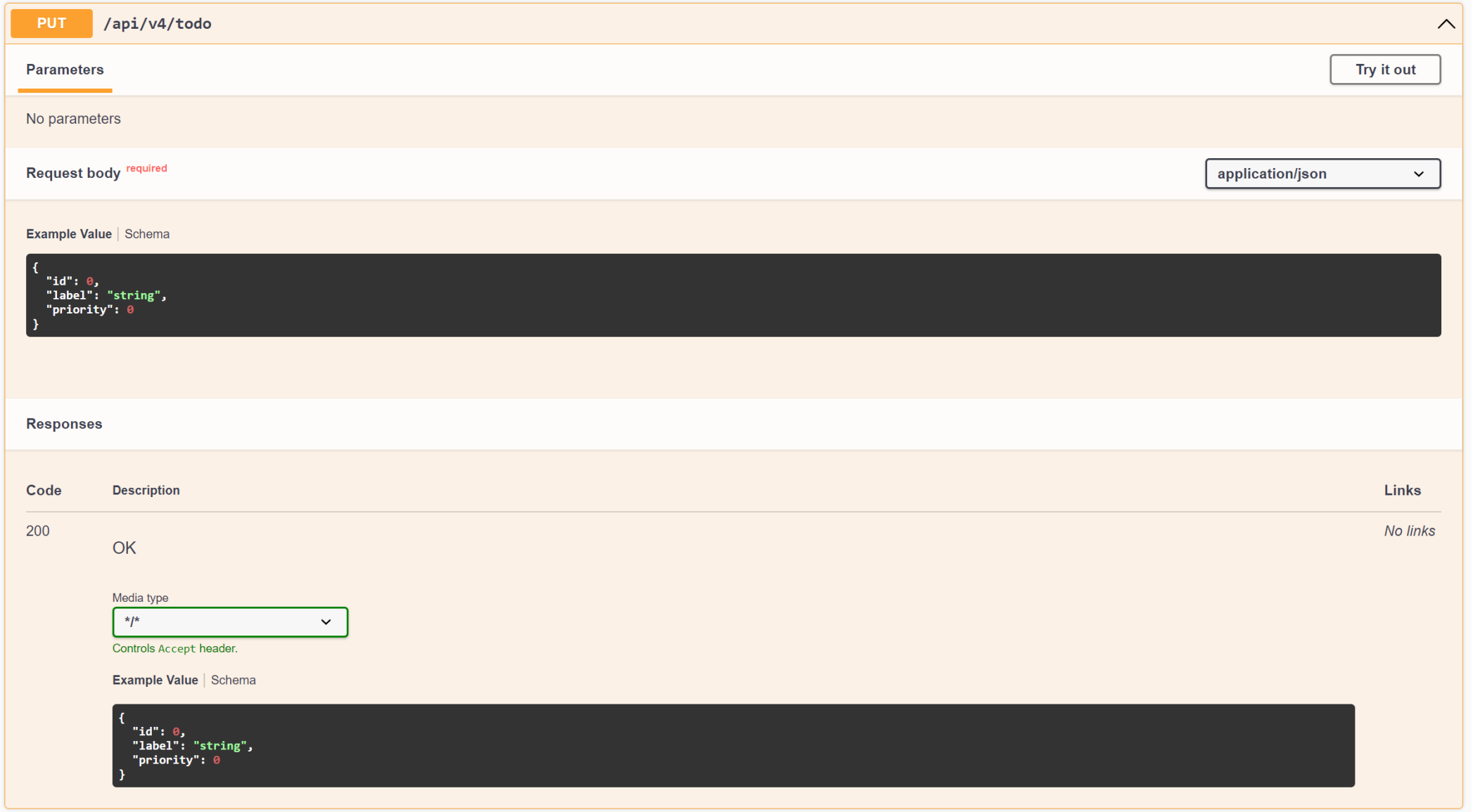
JSON api-docs
http://localhost:8080/v3/api-docs



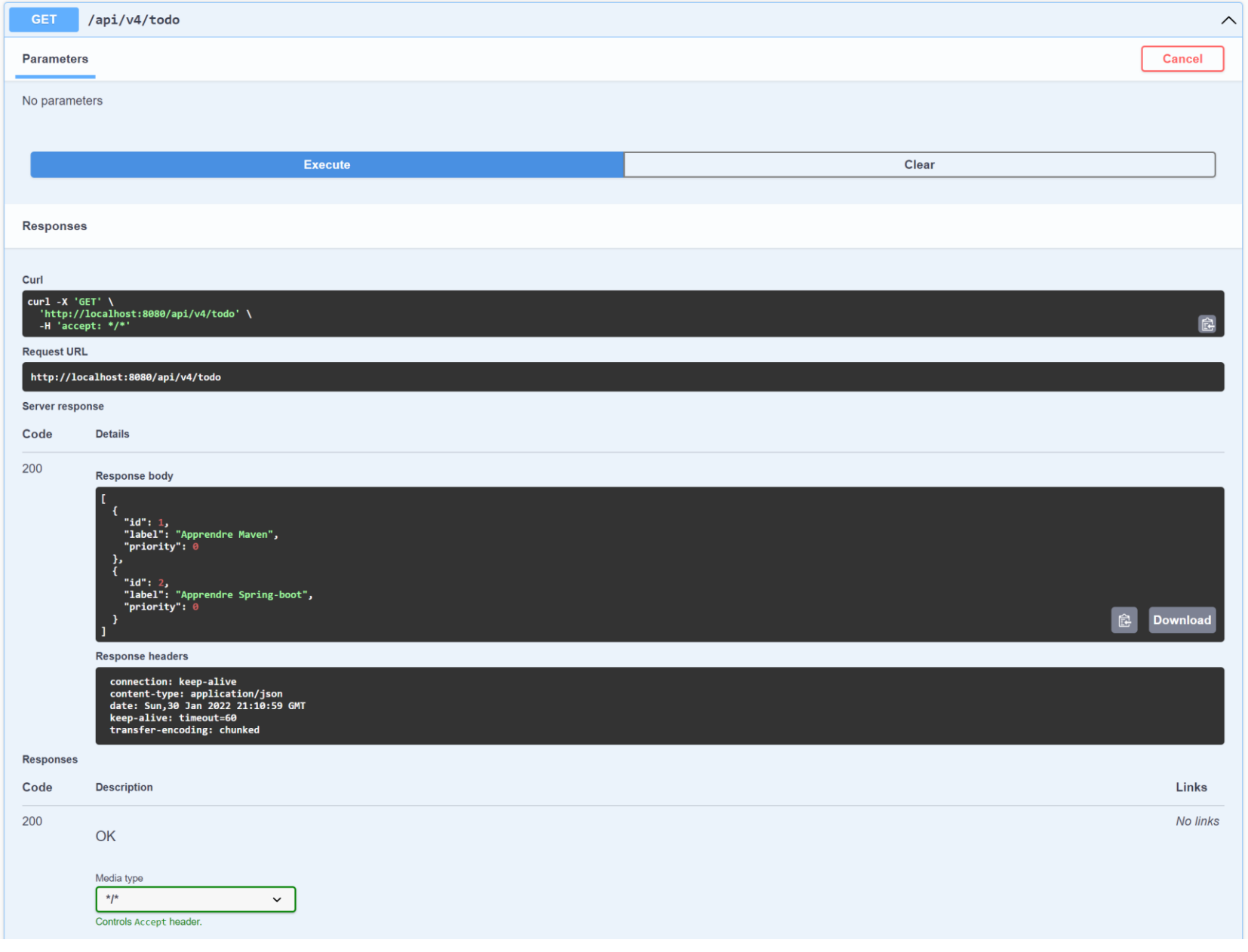
http://localhost:8080/swagger-ui.html



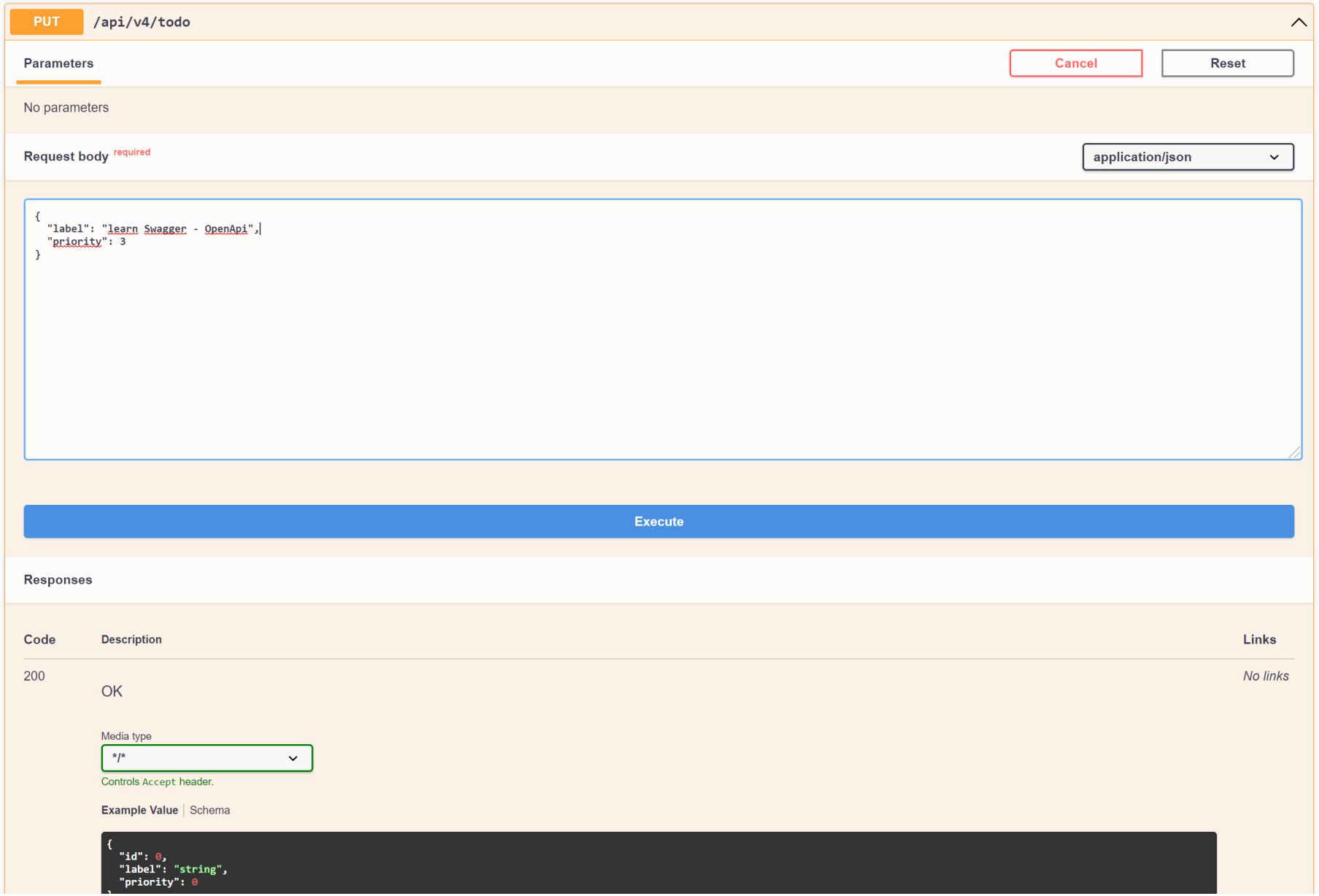
Detailed Rest operation



Execute Query



Example PUT with Json RequestBody



https://github.com/swagger-api/swagger-codegen

This is the Swagger Codegen project, which allows generation of API client libraries (SDK generation), server stubs and documentation automatically given an OpenAPI Spec. Currently, the following languages/frameworks are supported:

- **API clients**: **ActionScript**, **Ada**, **Apex**, **Bash**, **C#** (.net 2.0, 3.5 or later), **C++** (cpprest, Qt5, Tizen), **Clojure**, **Dart**, **Elixir**, **Elm**, **Eiffel**, **Erlang**, **Go**, **Groovy**, **Haskell** (http-client, Servant), **Java** (Jersey1.x, Jersey2.x, OkHttp, Retrofit1.x, Retrofit2.x, Feign, RestTemplate, RESTEasy, Vertx, Google API Client Library for Java, Rest-assured), **Kotlin**, **Lua**, **Node.js** (ES5, ES6, AngularJS with Google Closure Compiler annotations) **Objective-C**, **Perl**, **PHP**, **PowerShell**, **Python**, **R**, **Ruby**, **Rust** (rust, rust-server), **Scala** (akka, http4s, swagger-async-httpclient), **Swift** (2.x, 3.x, 4.x, 5.x), **Typescript** (Angular1.x, Angular2.x, Fetch, jQuery, Node)
- **Server stubs**: **Ada**, **C#** (ASP.NET Core, NancyFx), **C++** (Pistache, Restbed), **Erlang**, **Go**, **Haskell** (Servant), **Java** (MSF4J, Spring, Undertow, JAX-RS: CDI, CXF, Inflector, RestEasy, Play Framework, PKMST), **Kotlin**, **PHP** (Lumen, Slim, Silex, Symfony, Zend Expressive), **Python** (Flask), **NodeJS**, **Ruby** (Sinatra, Rails5), **Rust** (rust-server), **Scala** (Finch, Lagom, Scalatra)
- **API documentation generators**: **HTML**, **Confluence Wiki**
- **Configuration files**: **Apache2**
- **Others**: **JMeter**

Swagger CodeGen maven-plugin

```
<plugin>
    <groupId>io.swagger.codegen.v3</groupId>
    <artifactId>swagger-codegen-maven-plugin</artifactId>
    <version>${swagger-codegen.version}</version>

    <configuration>
        <inputSpec>http://localhost:8080/v3/api-docs</inputSpec>
    </configuration>
    <executions>
        <execution>
            <id>generate-swagger-typescript-angular-7</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <language>typescript-angular</language>
                <output>${basedir}/target/generated-typescript-angular7</output>
                <configOptions>
                    <ngVersion>7.2.12</ngVersion>
                </configOptions>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Example: Generate typescript Angular

mvn generate-sources

```
$ mvn -Pswagger-gen generate-sources
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< fr.an.tests:test-springboot-cruds >------------------
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- swagger-codegen-maven-plugin:3.0.32:generate (generate-swagger-typescript-angular-7) @ test-springboot-cruds ---
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\model\todoDTO.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\api\todoRestController.service.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\model\models.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\api\api.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\index.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\api.module.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\configuration.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\variables.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\encoder.ts
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\.gitignore
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\.npmignore
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\git_push.sh
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\ng-package.json
[INFO] writing file C:\arn\test-springboot-cruds\target\generated-typescript-angular7\.swagger-codegen\VERSION
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.609 s
[INFO] Finished at: 2022-01-30T22:21:21+01:00
[INFO] ------------------------------------------------------------------------
```

Generated Code (example: Typescript -Angular)

```
/**
 * OpenAPI definition
 * No description provided (generated by Swagger Codegen https://github.com/swagger-api/swagger-codegen)
 *
 * OpenAPI spec version: v0
 *
 *
 * NOTE: This class is auto generated by the swagger code generator program.
 * https://github.com/swagger-api/swagger-codegen.git
 * Do not edit the class manually.
 *//* tslint:disable:no-unused-variable member-ordering */

import { Inject, Injectable, Optional }                      from '@angular/core';
import { HttpClient, HttpHeaders, HttpParams,
         HttpResponse, HttpEvent }                           from '@angular/common/http';
import { CustomHttpUrlEncodingCodec }                        from '../encoder';

import { Observable }                                        from 'rxjs';

import { ResponseDTO } from '../model/responseDTO';
import { TodoDTO } from '../model/todoDTO';

import { BASE_PATH, COLLECTION_FORMATS }                     from '../variables';
import { Configuration }                                     from '../configuration';


@Injectable()
export class TodoRestControllerService {

    protected basePath = 'http://localhost:8080';
    public defaultHeaders = new HttpHeaders();
    public configuration = new Configuration();

    constructor(protected httpClient: HttpClient, @Optional()@Inject(BASE_PATH) basePath: string, @Optional() confi
        if (basePath) {
            this.basePath = basePath;
        }
        if (configuration) {
            this.configuration = configuration;
            this.basePath = basePath || configuration.basePath || this.basePath;
        }
    }
```

```
public putTodo4(body: TodoDTO, observe?: 'body', reportProgress?: boolean): Observable<TodoDTO>;
public putTodo4(body: TodoDTO, observe?: 'response', reportProgress?: boolean): Observable<HttpResponse<TodoDTO
public putTodo4(body: TodoDTO, observe?: 'events', reportProgress?: boolean): Observable<HttpEvent<TodoDTO>>;
public putTodo4(body: TodoDTO, observe: any = 'body', reportProgress: boolean = false ): Observable<any> {

    if (body === null || body === undefined) {
        throw new Error('Required parameter body was null or undefined when calling putTodo4.');
    }

    let headers = this.defaultHeaders;

    // to determine the Accept header
    let httpHeaderAccepts: string[] = [
        '*/*'
    ];
    const httpHeaderAcceptSelected: string | undefined = this.configuration.selectHeaderAccept(httpHeaderAccept
    if (httpHeaderAcceptSelected != undefined) {
        headers = headers.set('Accept', httpHeaderAcceptSelected);
    }

    // to determine the Content-Type header
    const consumes: string[] = [
        'application/json'
    ];
    const httpContentTypeSelected: string | undefined = this.configuration.selectHeaderContentType(consumes);
    if (httpContentTypeSelected != undefined) {
        headers = headers.set('Content-Type', httpContentTypeSelected);
    }

    return this.httpClient.request<TodoDTO>('put',`${this.basePath}/api/todo`,
        {
            body: body,
            withCredentials: this.configuration.withCredentials,
            headers: headers,
            observe: observe,
            reportProgress: reportProgress
        }
    );
}
```

Another Example : generate Java

```
<plugin>
    <groupId>io.swagger.codegen.v3</groupId>
    <artifactId>swagger-codegen-maven-plugin</artifactId>
    <version>${swagger-codegen.version}</version>
    <configuration>
        <inputSpec>http://localhost:8080/v3/api-docs</inputSpec>
    </configuration>
    <executions>
        <execution>
            <id>generate-java</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <language>java</language>
                <output>${basedir}/target/generated-java</output>
            </configuration>
        </execution>
        <execution>
            <id>generate-java-retrofit2</id>
            <phase>generate-sources</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <language>java</language>
                <library>retrofit2</library>
                <output>${basedir}/target/generated-java-retrofit2</output>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Java
Default library=OkHttp

Java
+ library=Retrofit2

Generated Code (Java, default using OkHttp)

```java
package io.swagger.client.api;

import io.swagger.client.ApiCallback;

public class TodoRestControllerApi {
    private ApiClient apiClient;

    public TodoRestControllerApi() {
        this(Configuration.getDefaultApiClient());
    }

    public TodoRestControllerApi(ApiClient apiClient) {
        this.apiClient = apiClient;
    }
```

```java
     * Build call for putTodo4
    public com.squareup.okhttp.Call putTodo4Call(TodoDTO body, final ProgressResponseBody.ProgressListener progress
        Object localVarPostBody = body;

         / create path and map variables
         tring localVarPath = "/api/todo";

         ist<Pair> localVarQueryParams = new ArrayList<Pair>();
         ist<Pair> localVarCollectionQueryParams = new ArrayList<Pair>();

         ap<String, String> localVarHeaderParams = new HashMap<String, String>();

         ap<String, Object> localVarFormParams = new HashMap<String, Object>();

         final String[] localVarAccepts = {
           "*/*"
         ;
         final String localVarAccept = apiClient.selectHeaderAccept(localVarAccepts);
         f (localVarAccept != null) localVarHeaderParams.put("Accept", localVarAccept);

         final String[] localVarContentTypes = {
           "application/json"
         ;
         final String localVarContentType = apiClient.selectHeaderContentType(localVarContentTypes);
        localVarHeaderParams.put("Content-Type", localVarContentType);

        if(progressListener != null) {
            apiClient.getHttpClient().networkInterceptors().add(new com.squareup.okhttp.Interceptor() {
                @Override
                public com.squareup.okhttp.Response intercept(com.squareup.okhttp.Interceptor.Chain chain) throws I
                    com.squareup.okhttp.Response originalResponse = chain.proceed(chain.request());
                    return originalResponse.newBuilder()
                    .body(new ProgressResponseBody(originalResponse.body(), progressListener))
                    .build();
                }
            });
        }

        String[] localVarAuthNames = new String[] {  };
        return apiClient.buildCall(localVarPath, "PUT", localVarQueryParams, localVarCollectionQueryParams, localVa
    }
```

Generated Code (Java + Retrofit2)

```
public ApiClient(String authName, String clientId, String secret, String user
  this(authName);
  this.getTokenEndPoint()
    .setClientId(clientId)
    .setClientSecret(secret)
    .setUsername(username)
    .setPassword(password);
}

public void createDefaultAdapter() {
  json = new JSON();
  okBuilder = new OkHttpClient.Builder();

  String baseUrl = "http://localhost:8080";
  if (!baseUrl.endsWith("/"))
    baseUrl = baseUrl + "/";

  adapterBuilder = new Retrofit
    .Builder()
    .baseUrl(baseUrl)
    .addConverterFactory(ScalarsConverterFactory.create())
    .addConverterFactory(GsonCustomConverterFactory.create(json.getGson()));
}

public <S> S createService(Class<S> serviceClass) {
  return adapterBuilder
    .client(okBuilder.build())
    .build()
    .create(serviceClass);
}
```

```
import retrofit2.Call;
import retrofit2.http.*;

import okhttp3.RequestBody;
import okhttp3.ResponseBody;

import io.swagger.client.model.ResponseDTO;
import io.swagger.client.model.TodoDTO;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public interface TodoRestControllerApi {
   * @param id  (required)
  @DELETE("api/todo/{id}")
  Call<TodoDTO> deleteTodo4(
            @retrofit2.http.Path("id") Integer id
  );

   * @param id  (required)
  @GET("api/todo/{id}")
  Call<TodoDTO> get5(
            @retrofit2.http.Path("id") Integer id
  );

   * @return Call&lt;List&lt;TodoDTO&gt;&gt;
  @GET("api/todo")
  Call<List<TodoDTO>> list4();


   * @param body  (required)
  @Headers({
    "Content-Type:application/json"
  })
  @POST("api/todo")
  Call<ResponseDTO> postTodo4(
                    @retrofit2.http.Body TodoDTO body
  );
```

Sample Retrofit2 Java Client

```
public void testSwaggerCliRetrofit2() {
    ApiClient apiClient = new ApiClient();
    TodoRestControllerApi svc = apiClient.createService(TodoRestControllerApi.class);

    Call<List<TodoDTO>> call = svc.list4();
    List<TodoDTO> res = executeGetBody(call, "GET /api/todo");

    System.out.println("http GET /api/todo => got " + res.size() + " elts");
}

private static <T> T executeGetBody(Call<T> call, String callDescr) {
    Response<T> resp;
    try {
        resp = call.execute();
    } catch (IOException ex) {
        throw new RuntimeException("can not call http " + callDescr, ex);
    }
    if (! resp.isSuccessful()) {
        throw new RuntimeException("Failure in http " + callDescr + ", " +
                "code:" + resp.code() + " message:" + resp.message());
    }
    return resp.body();
}
```

Exposing DTO + API using Internal Entity

Mapping … entity2Dto / dto2Entity

Naive (Not working) @Entity to JSON

```java
@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 class maybe not JSON compatible

Entity object invalid After transaction commit (managed lifecycle)

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

```java
@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

```java
@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

```java
@Data
public class UserLightDTO {

    public String firstName;
    public String lastName;
    public String pseudo;

}
```

Owner personal view

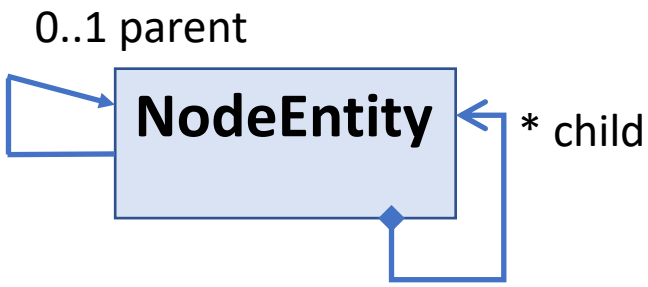
```java
@Data
public class SecuredUserDetailDTO {
    public String email;
    public String firstName;
    public String lastName;
    public String pseudo;

    // computed from group,settings, ...
    public List<String> grantedPermissions;
}
```

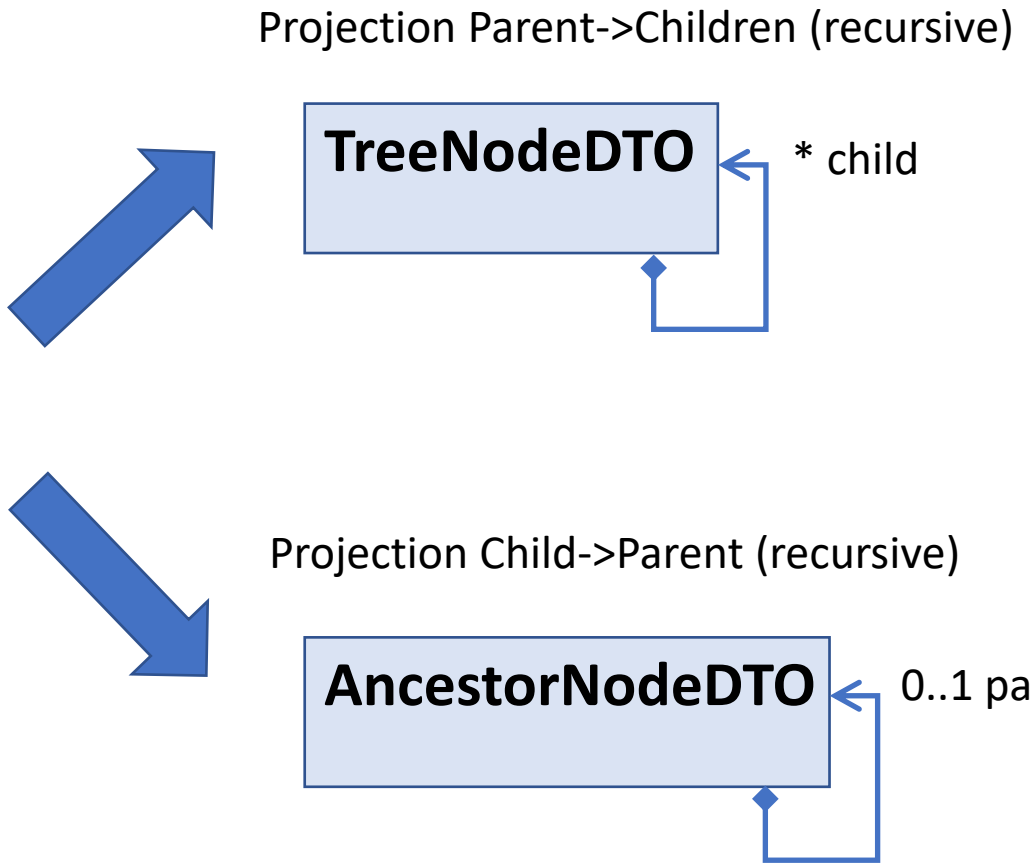
Public detailed view (high data volume)

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

Entity -> Projection DTOs = « Cutting » relations

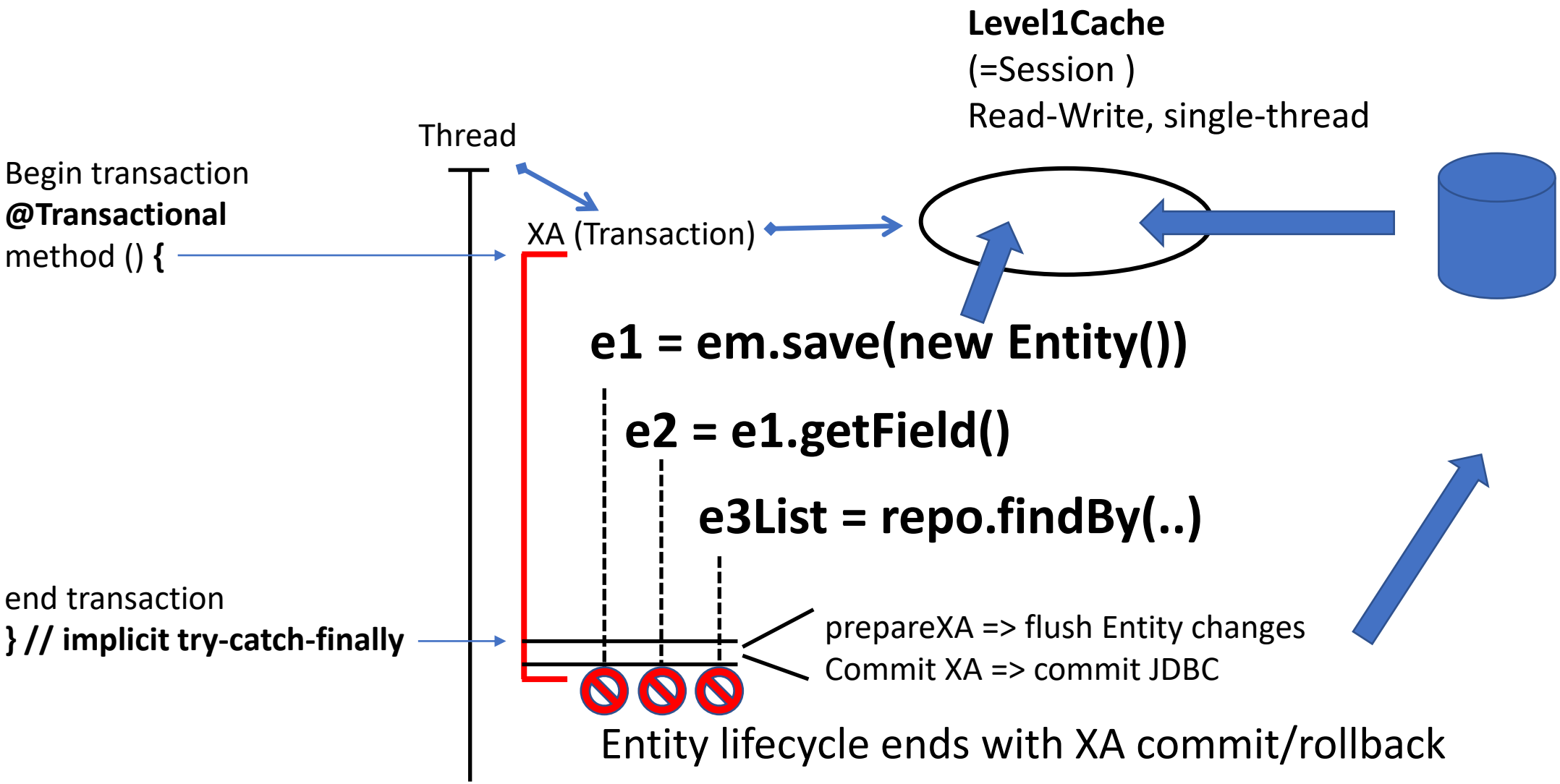
```java
@Entity
@Getter @Setter
public class NodeEntity {

    @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;

}
```



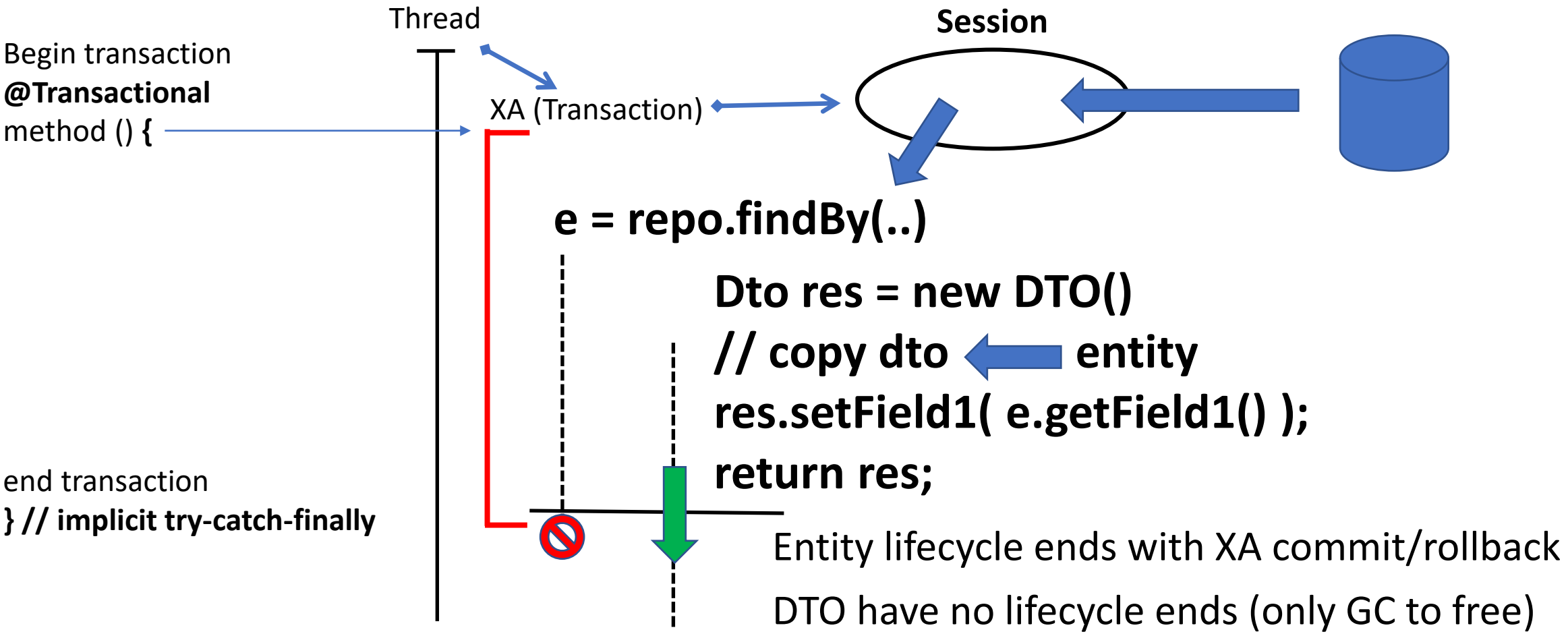


Reminder Part 2 … Entity lifecycle in Session

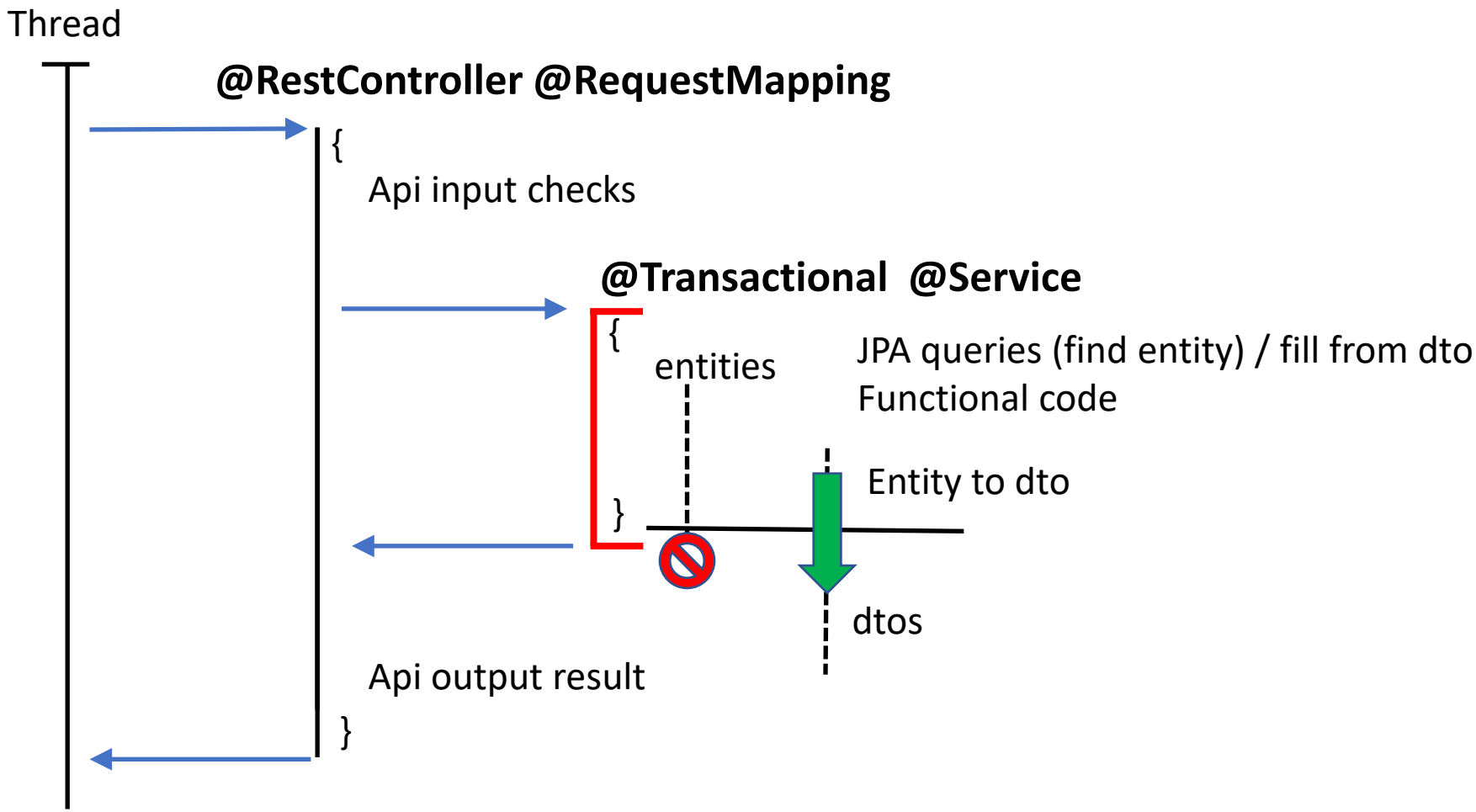
Entity : Lifecycle managed by Session (Transaction)



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 persiste
WARN  JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Th
INFO  o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/
```

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

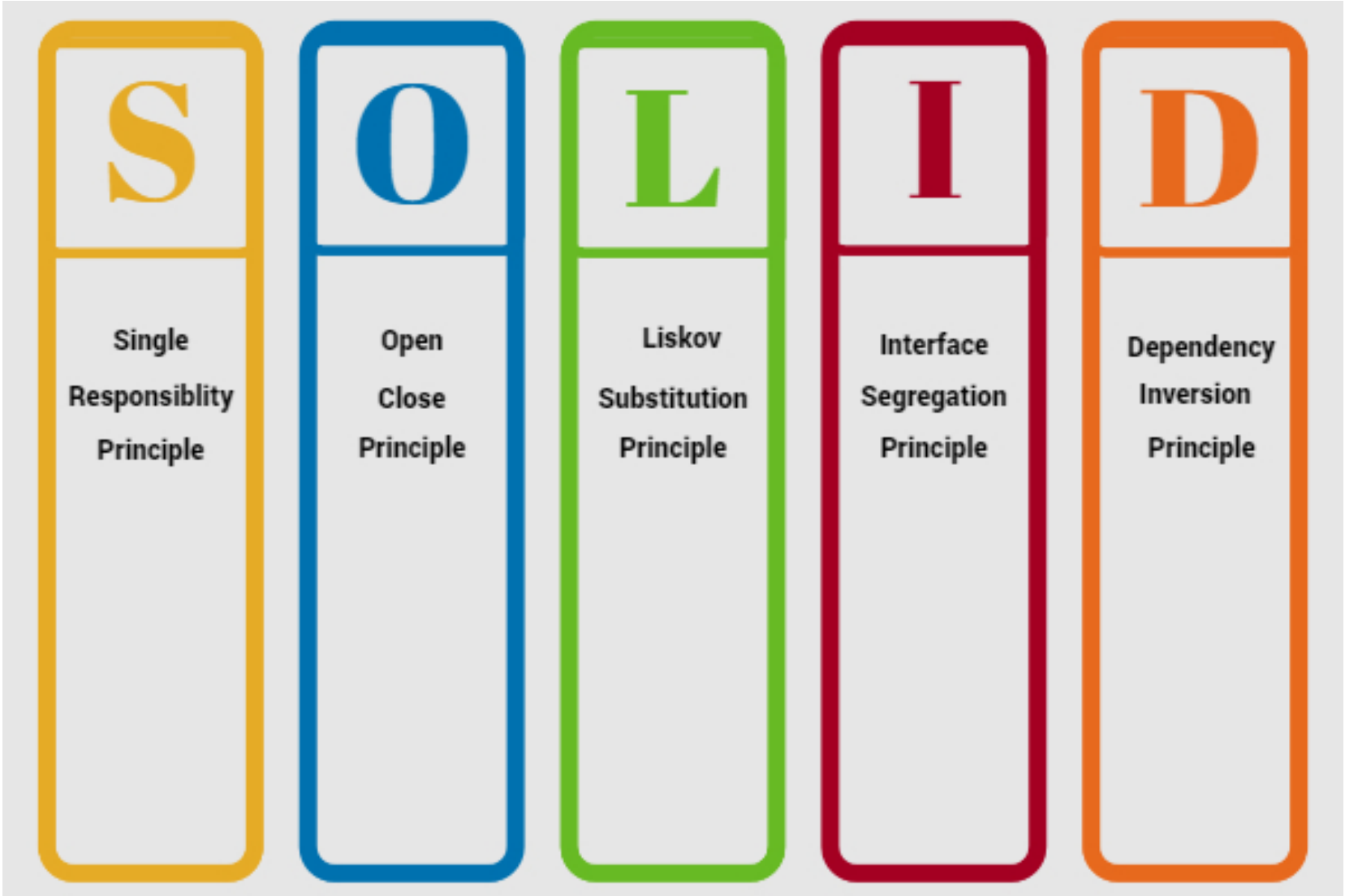
Bad default value in springboot

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: unmarshall, 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

```java
@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)

```java
@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 »

```java
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);
}
```

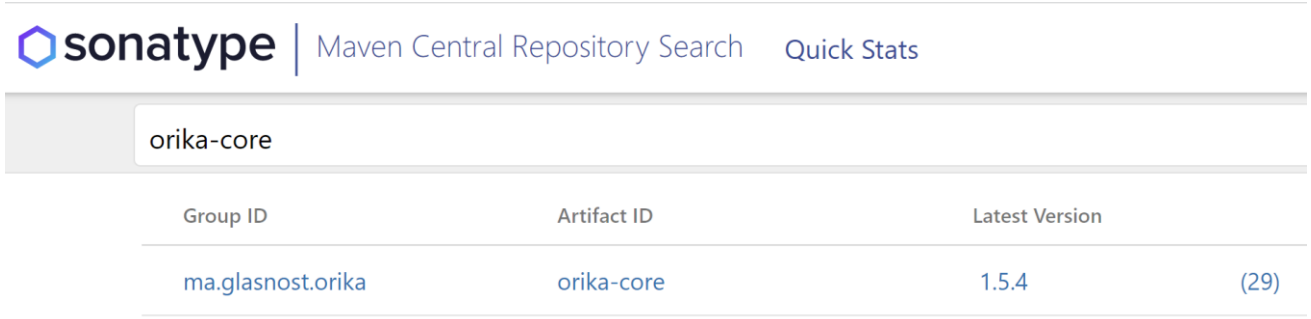
Using Orika MapperFacade

```
@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);
    }

}
```

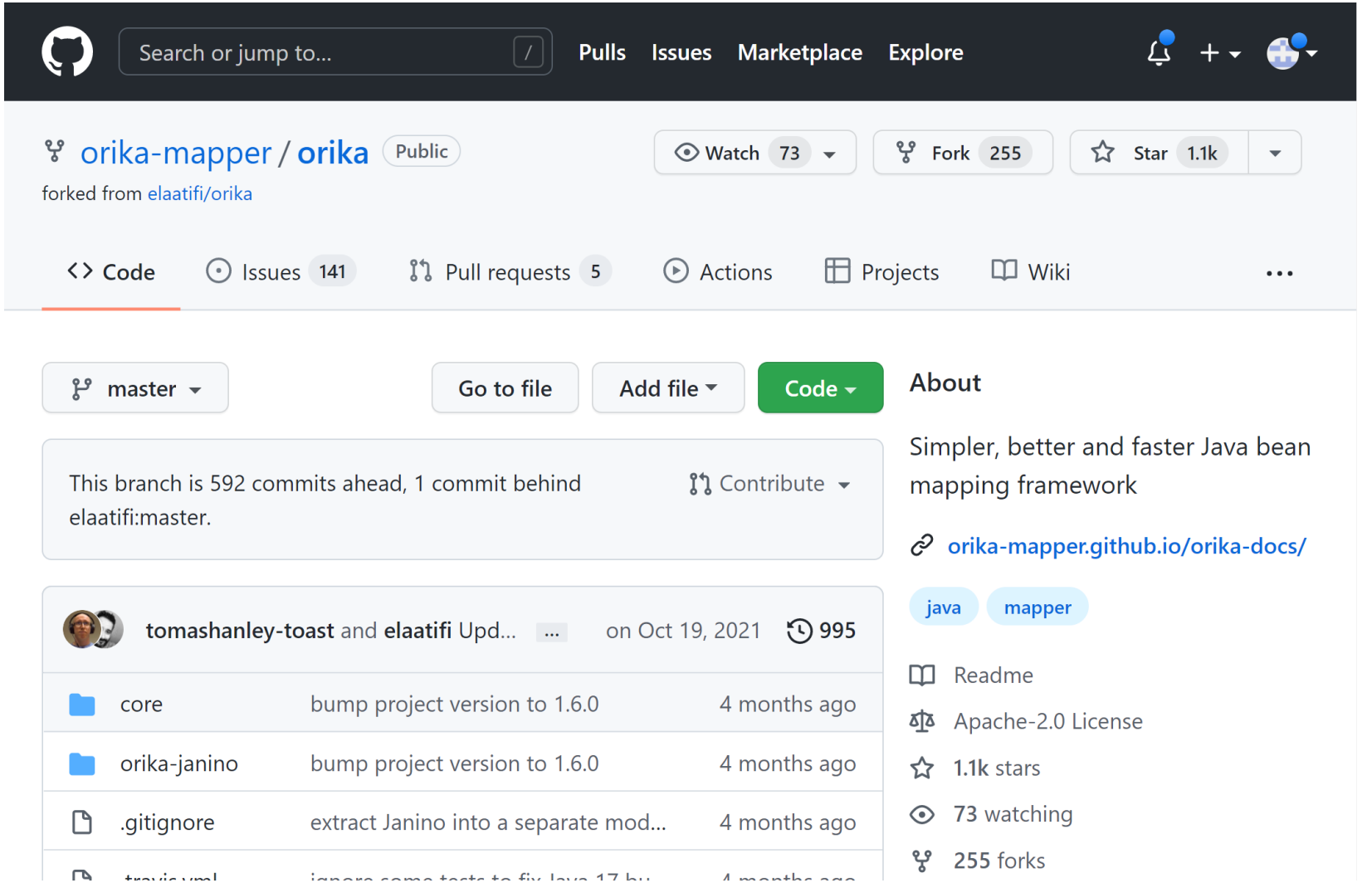
Orika

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

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



https://github.com/orika-mapper/orika



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

Orika *simpler, lighter and faster Java bean mapping*

User Guide

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

Orika Intro … customizing Class Mapper

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

Override custom
Bi-Directional field Mapping
= fieldAtoB + fieldBToA

Convention over configuration

```
.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

Default nested naming Convention !!!
« aaaBbb » <-> « aaa.bbb »

```
.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 {
```

Details

Naming convention for 1 Property

In-line property syntax:

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

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();
```

Details

Class to Class Full
Custom Mapper
(handle all fields)

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

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

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

Details

```
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
   }
}
```

Custom « new » Object Factory

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

Details

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

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

More customizations … Filter

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

Details

customize dynamic mapping behavior

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 !!!

DEFAULT: javaassist

ma.glasnost.orika.impl.generator.JavassistCompilerStrategy

ma.glasnost.orika.impl.generator.JaninoCompilerStrategy

strategy → Janino
(also in logback)

→ Opaque « .class » bytecode …
Default NO generated/debuggable « .java » Source

→ EclipseJdt

ma.glasnost.orika.impl.generator.EclipseJdtCompilerStrategy

add

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

- orika-core : 1.5.4 [compile]
  - javassist : 3.24.0-GA [compile]
  - paranamer : 2.8 [compile]
  - slf4j-api : 1.7.32 (managed from 1.7.25) [compile]
  - java-sizeof : 0.0.5 [compile]
  - janino : 3.1.6 (managed from 3.0.8) [compile]
    - commons-compiler : 3.1.6 [compile]

→ Debuggable
« .class » bytecode + generated « .java » Source-Code with Line numbers

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..

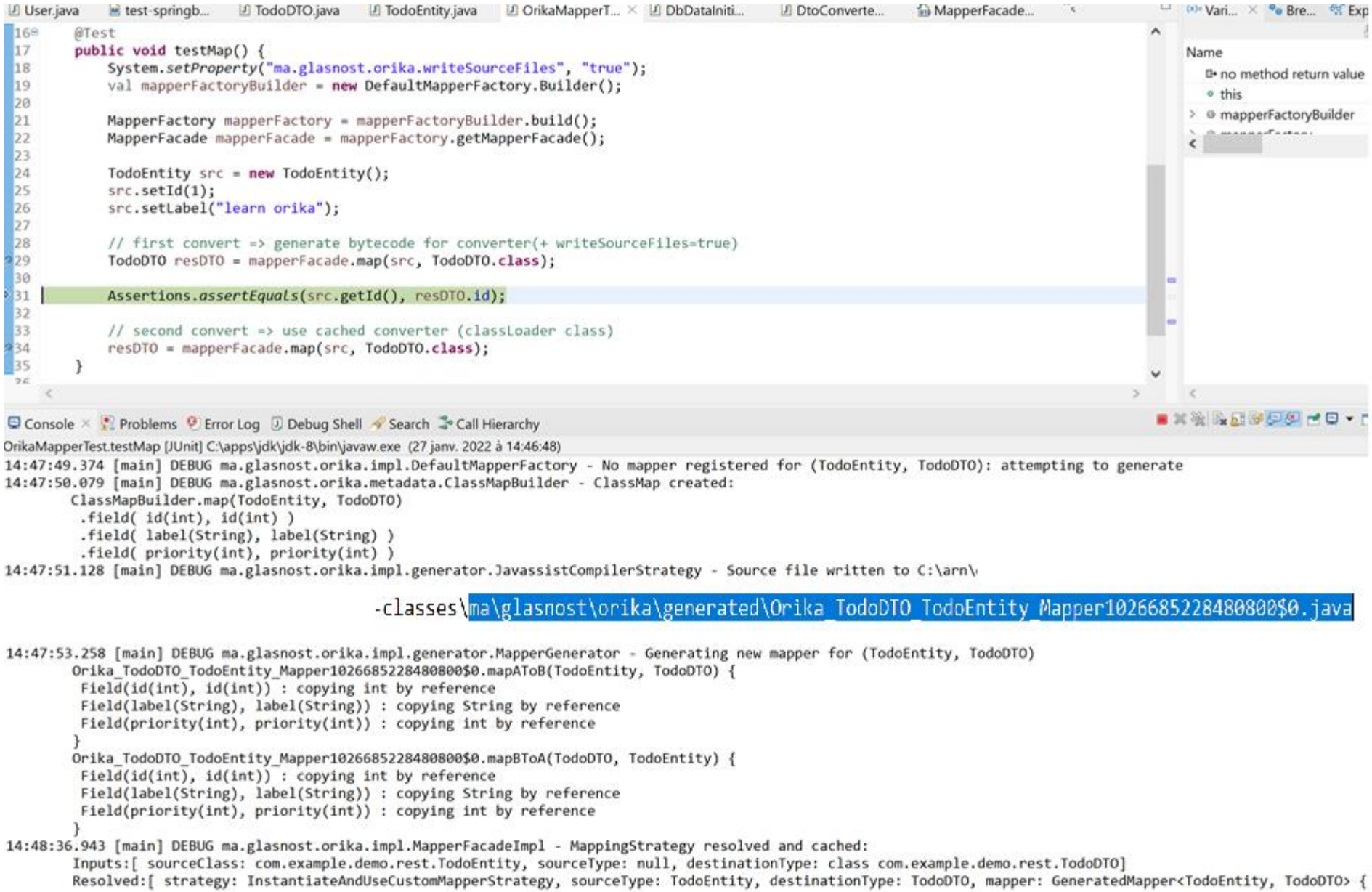
https://github.com/orika-mapper/orika/blob/master/core/src/main/java/ma/glasnost/orika/impl/generator/CompilerStrategy.java#L53

```
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)



```
User.java  test-springb...  TodoDTO.java  TodoEntity.java  OrikaMapperT...  DbDataIniti...  DtoConverte...  MapperFacade...

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    }

Vari...  Bre...  Exp
Name
no method return value
this
mapperFactoryBuilder

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_Mapper10266852284808800$0.java

14:47:53.258 [main] DEBUG ma.glasnost.orika.impl.generator.MapperGenerator - Generating new mapper for (TodoEntity, TodoDTO)
        Orika_TodoDTO_TodoEntity_Mapper10266852284808800$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_Mapper10266852284808800$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> {
```

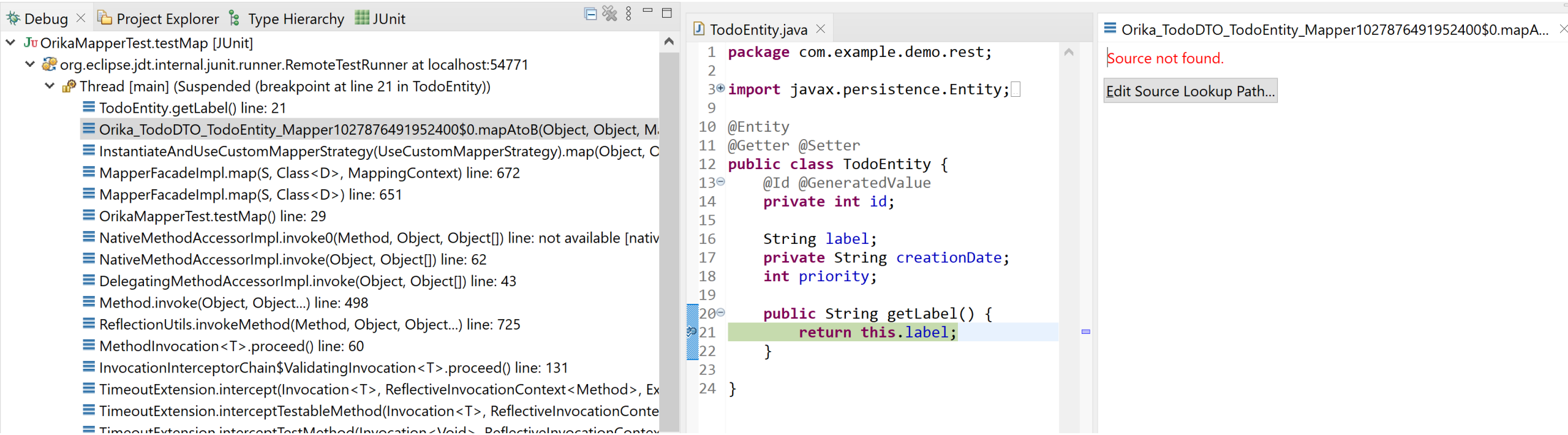
Generated code.. (file not visible in Eclipse?)

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

public class Orika_TodoDTO_TodoEntity_Mapper102668522848080O$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) {
```

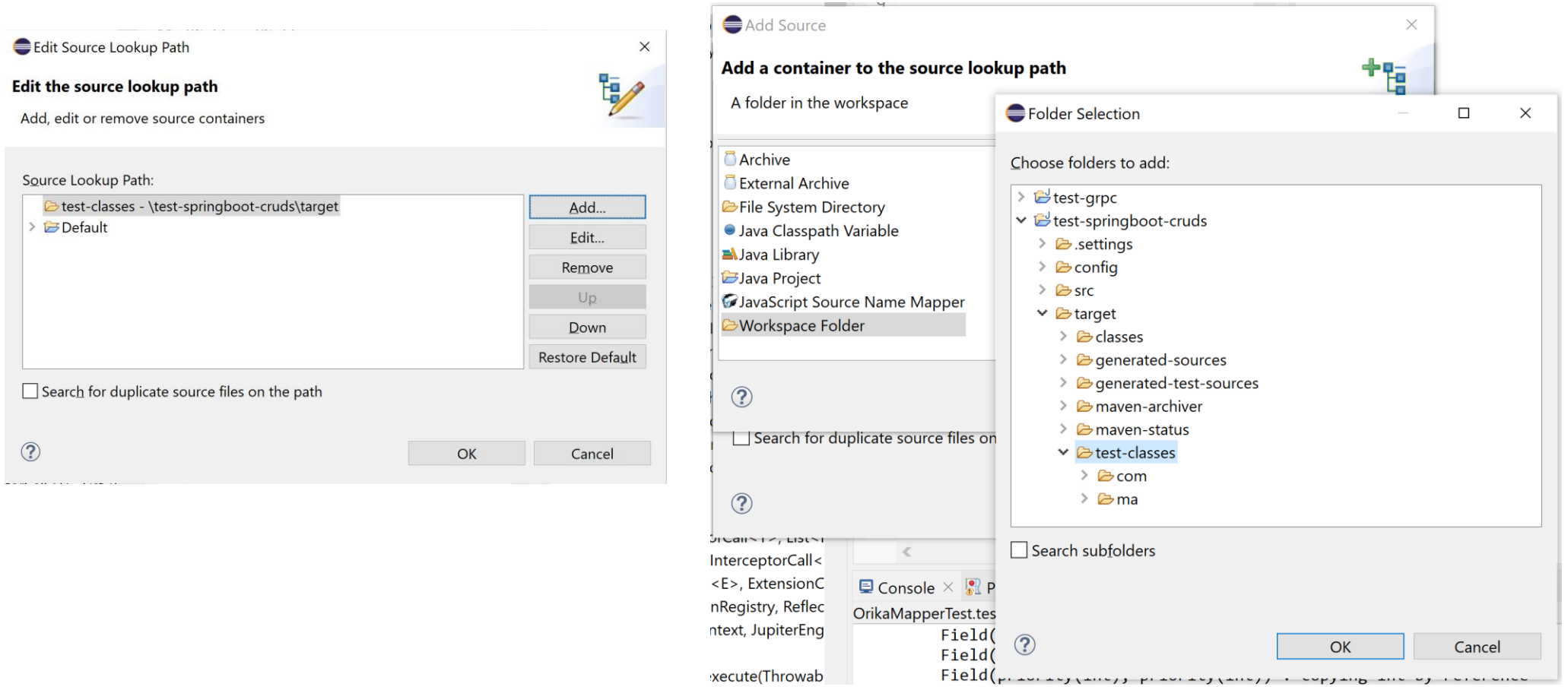
Optimal compiled code
Entity -> DTO
(with extra cast)

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

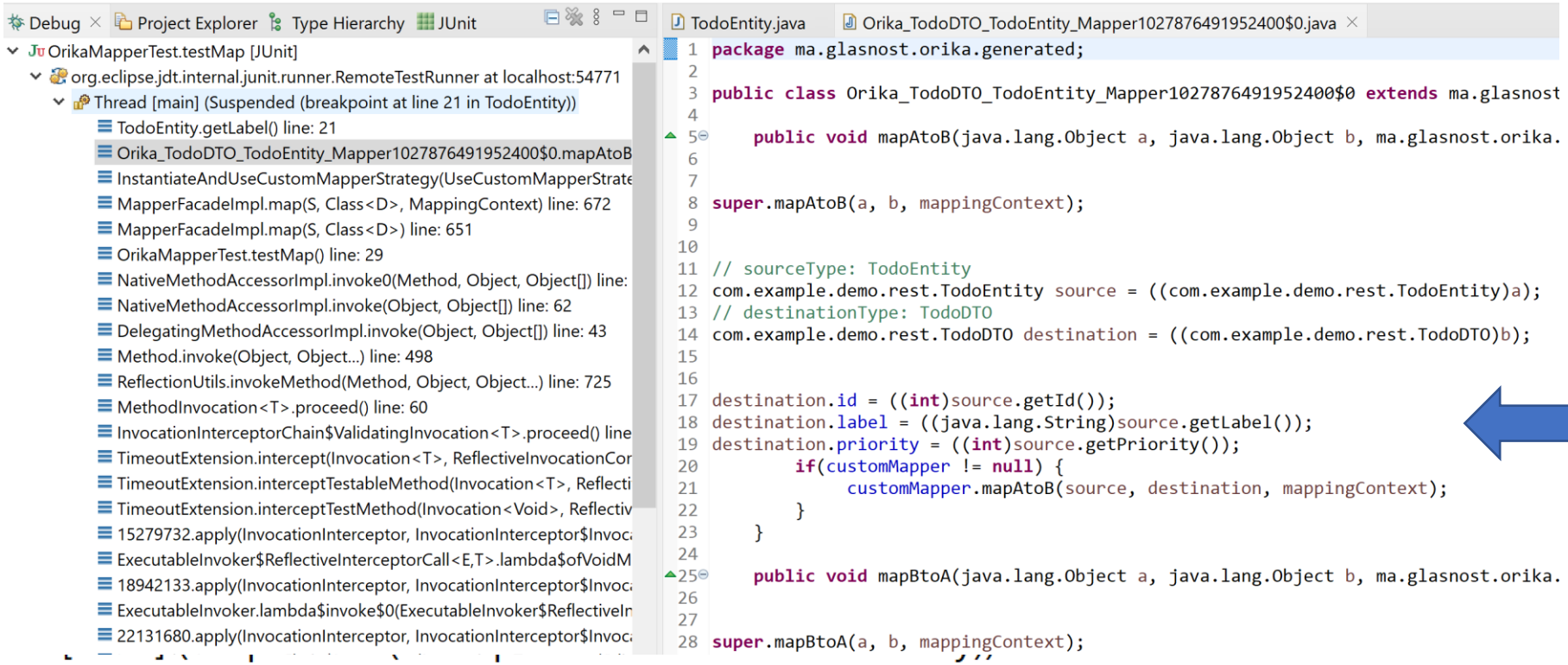
Generated Code .. Source not visible in Eclipse



Adding Source



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



Supposed to be here
« .getLabel() »
Line 18 … ??

TodoEntity.getLabel() line: 21

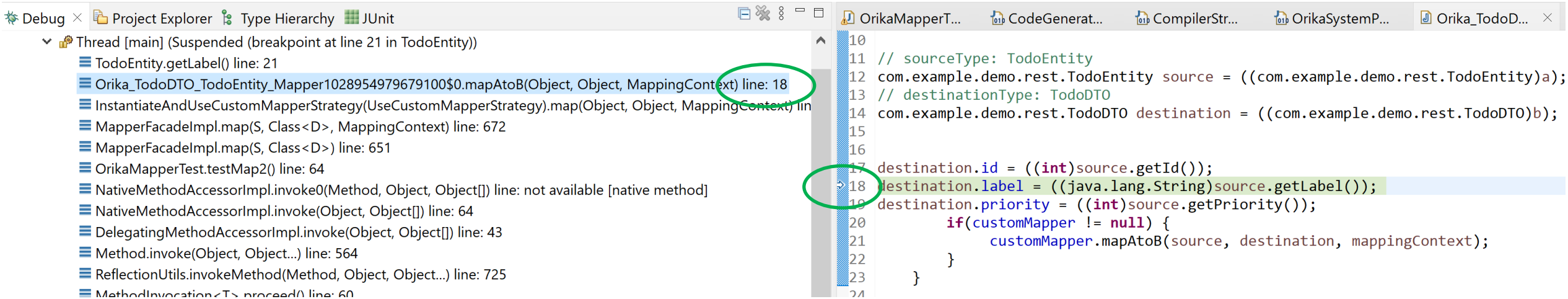
Orika_TodoDTO_TodoEntity_Mapper1027876491952400$0.mapAtoB(Object, Object, MappingContext) line: not available

InstantiateAndUseCustomMapperStrategy(UseCustomMapperStrategy).map(Object, Object, MappingContext) line: 77

MapperFacadeImpl.map(S, Class<D>, MappingContext) line: 672

Activating EclipseJdtCompilerStrategy

```java
@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();
```



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

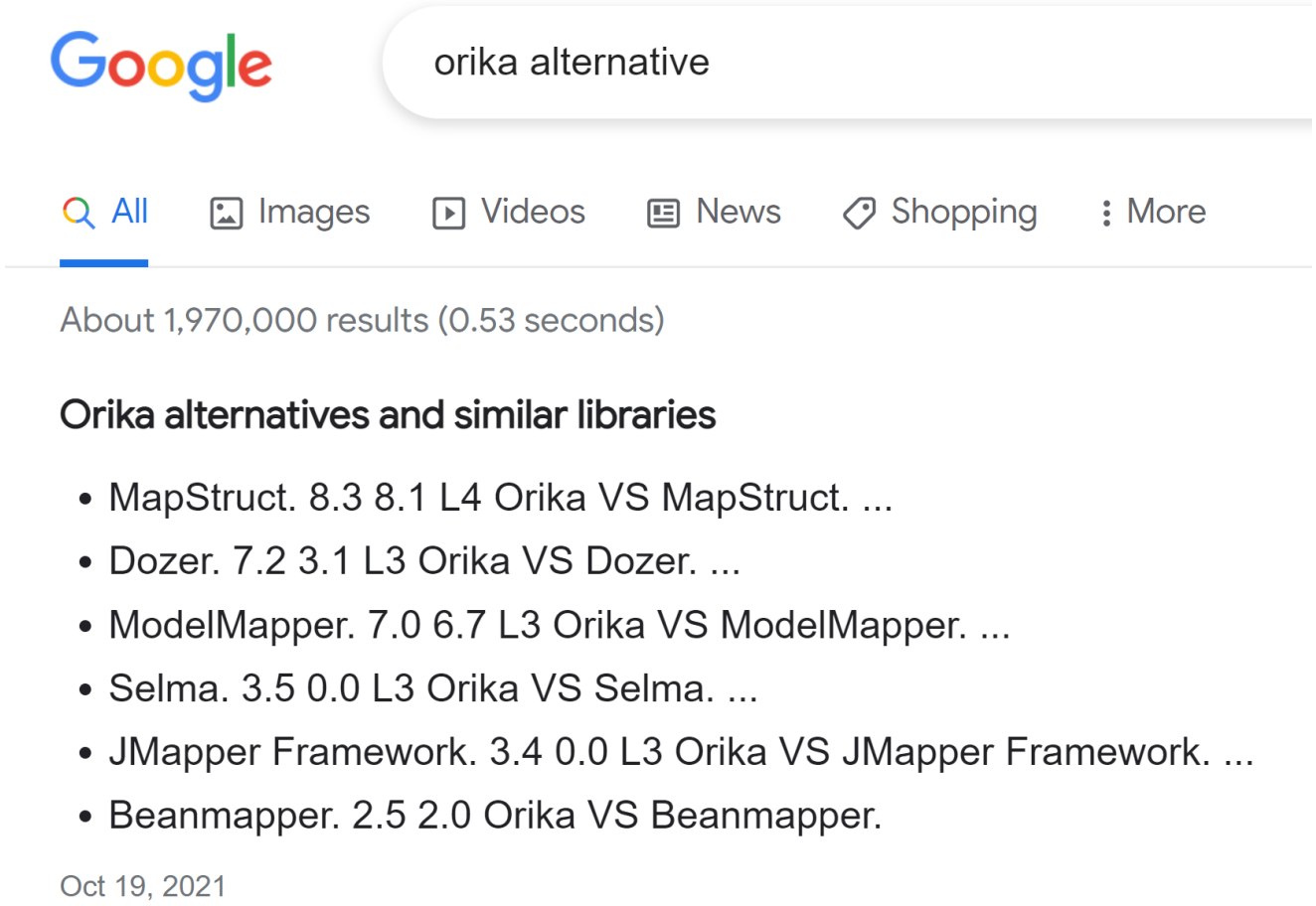
4/ No need to configure compile-time code generator / IDE plugins

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

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

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

Orika Alternatives



« **MapStruct** » chosen in JHipster

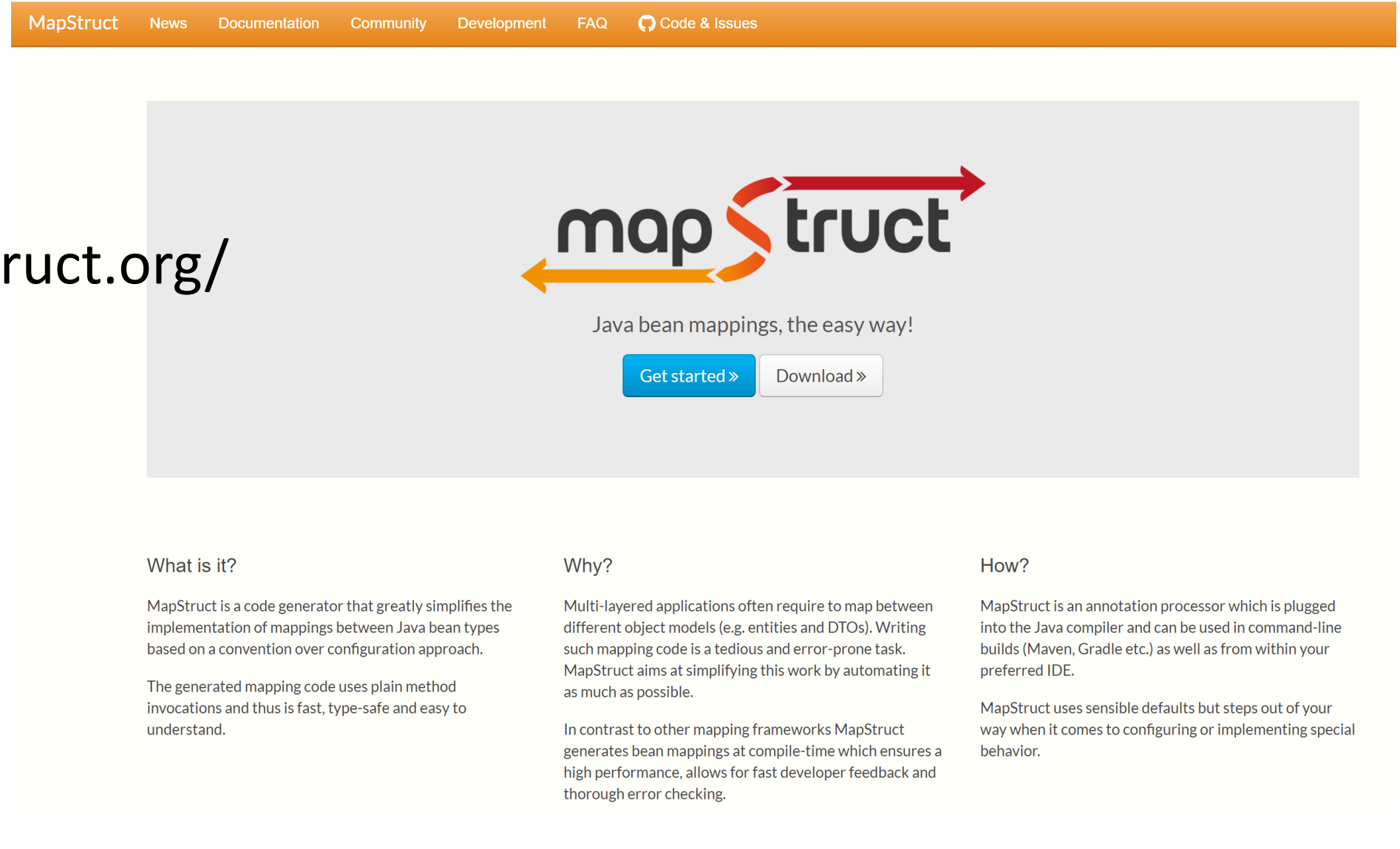
++ code generated at compile-time
-- … need IDE plugins / config
-- need to declare all methods signatures in interfaces

Non viable legacy ideas

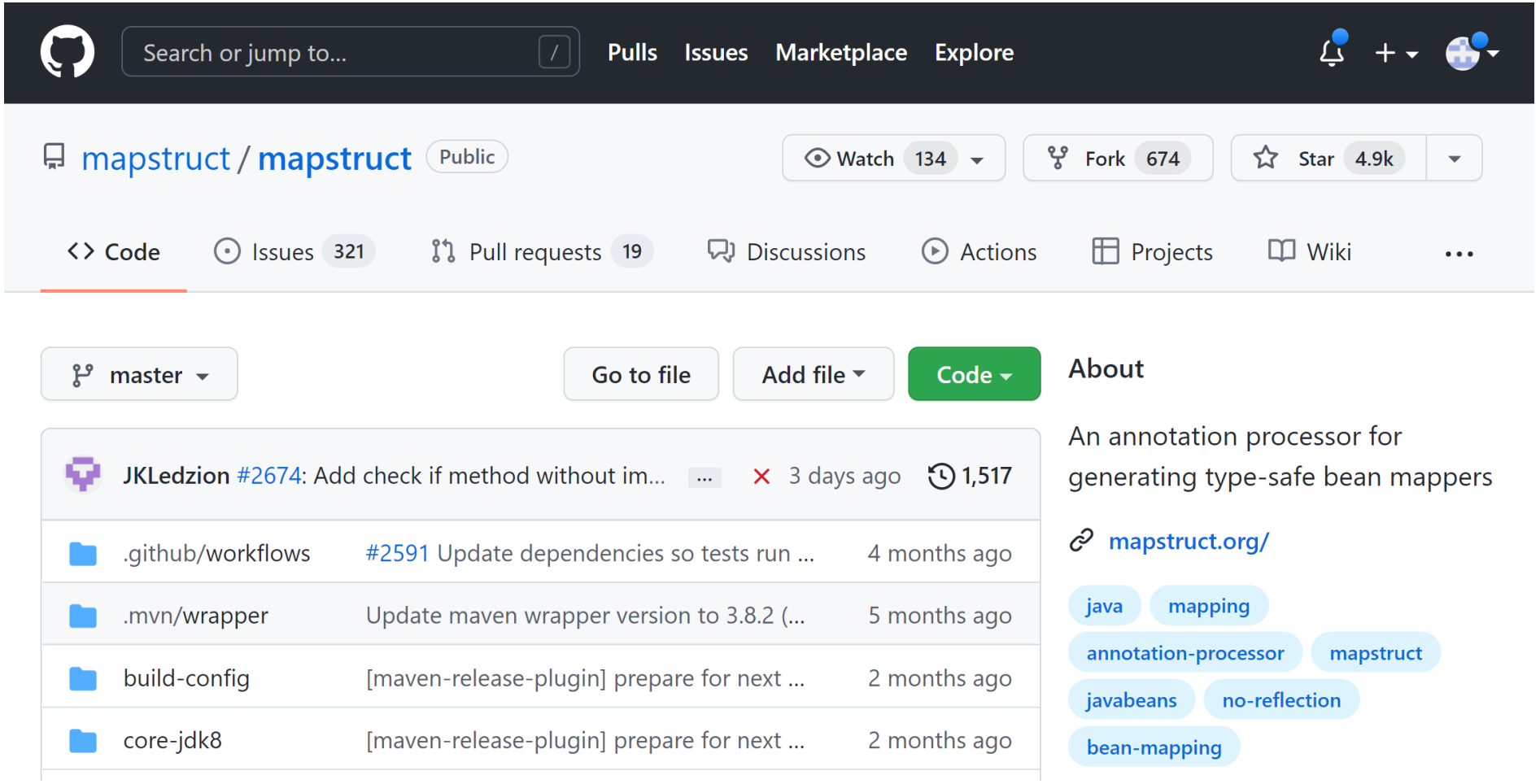
« **Dozer** » !!
Old, buggy, NO code generated
Very bad performances
Fully by introspection

MapStruct
(alternative chosen in Jhipster)

https://mapstruct.org/



https://github.com/mapstruct/mapstruct

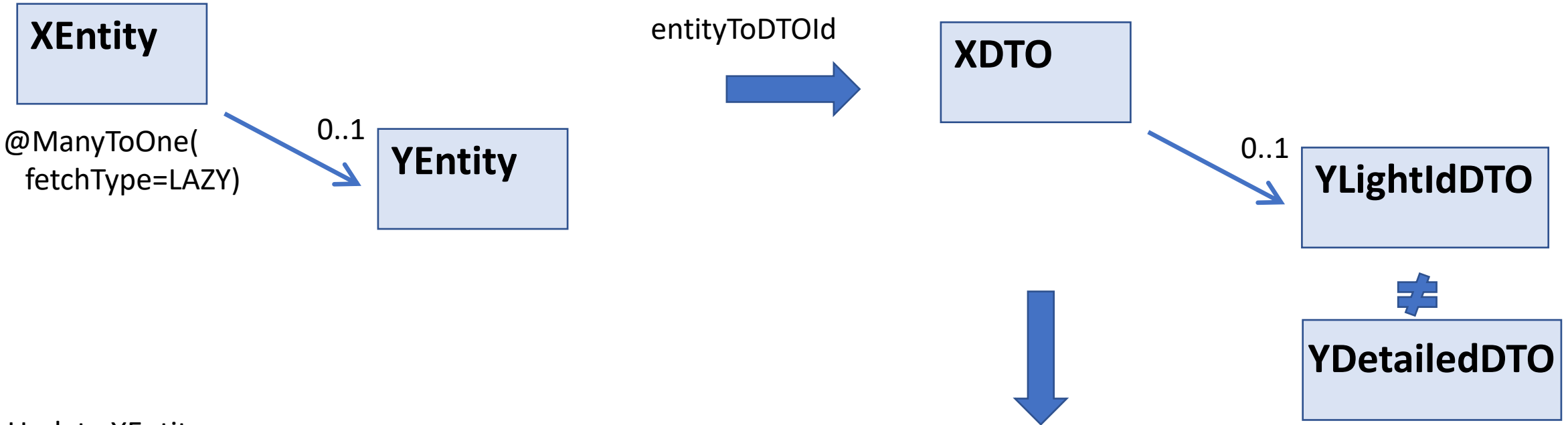


Orika … no findById in Springboot/JPA ?

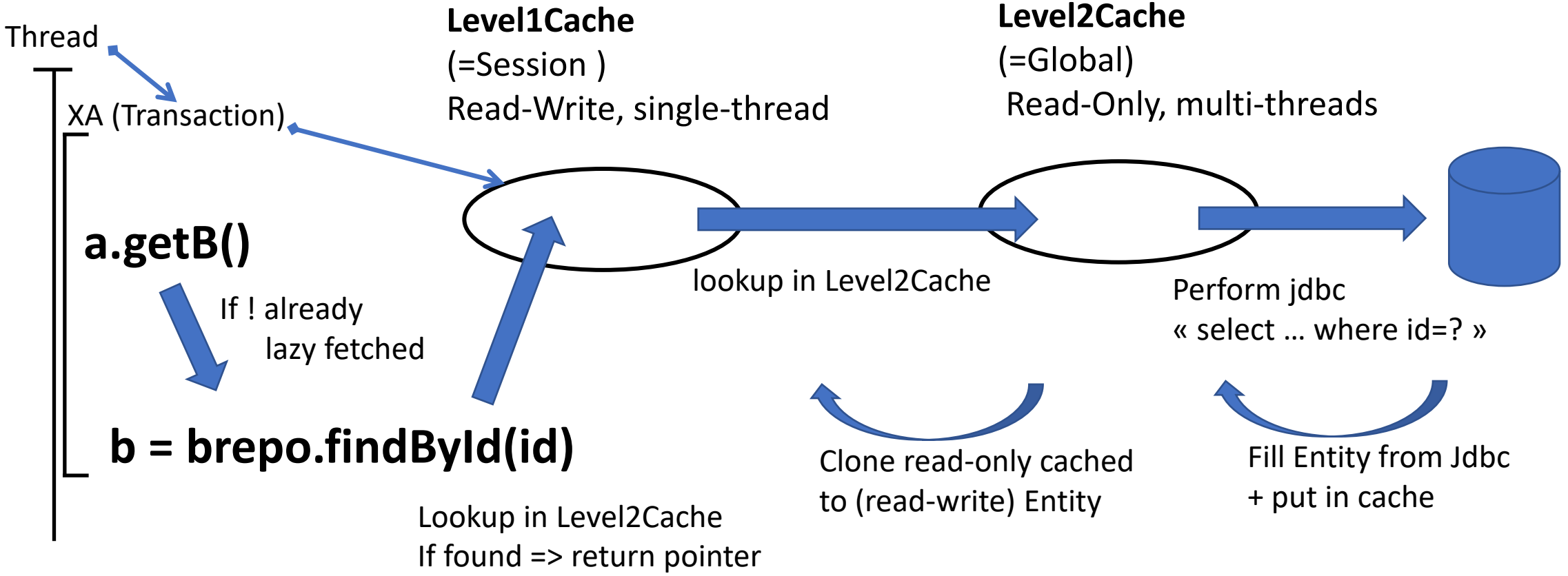
-- NO built-in recognized support for findById() « id » <-> « entity » in JPA + springboot

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() !!

```
public BadMergeSourceDTO create(BadMergeSourceDTO src) {
    BadMergeSourceEntity entity = new BadMergeSour
    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

Detached (invalid) entity

.save(entity);

.map(e

0 upda

Debu

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

```
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

… 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: PU
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$300(JdbcResourceLocalTransactionC
        at org.hibernate.resource.transaction.backend.jdbc.internal.JdbcResourceLocalTransactionCoordinatorImpl$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]
```

Correct mapper with findById

```java
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 EntityNotFound 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 = sourc
    sourceRepo.flu

    return dtoConv
}
```

- $$_hibernate_interceptor= ByteBuddyInterceptor (id=173)
- displayName= null
- field1= null
- field2= null
- field3= null
- field4= null
- field5= null
- id= 0

Do not trust fields … hibernate use « ByteBuddyInterceptor » + store in handler

Getting correct 404 for invalid id

```java
@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"}
```

```
< HTTP/1.1 404
```

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 : c
        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.j
        at org.hibernate.event.internal.AbstractFlushingEventListener.prepareEntityFlushes(AbstractFlushingEventListener.java:149) ~[hibernate-core-5.6.1.F
        at org.hibernate.event.internal.AbstractFlushingEventListener.flushEverythingToExecutions(AbstractFlushingEventListener.java:82) ~[hibernate-core-5
        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.Fina
        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() « id » <-> « 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 transfering useless network data

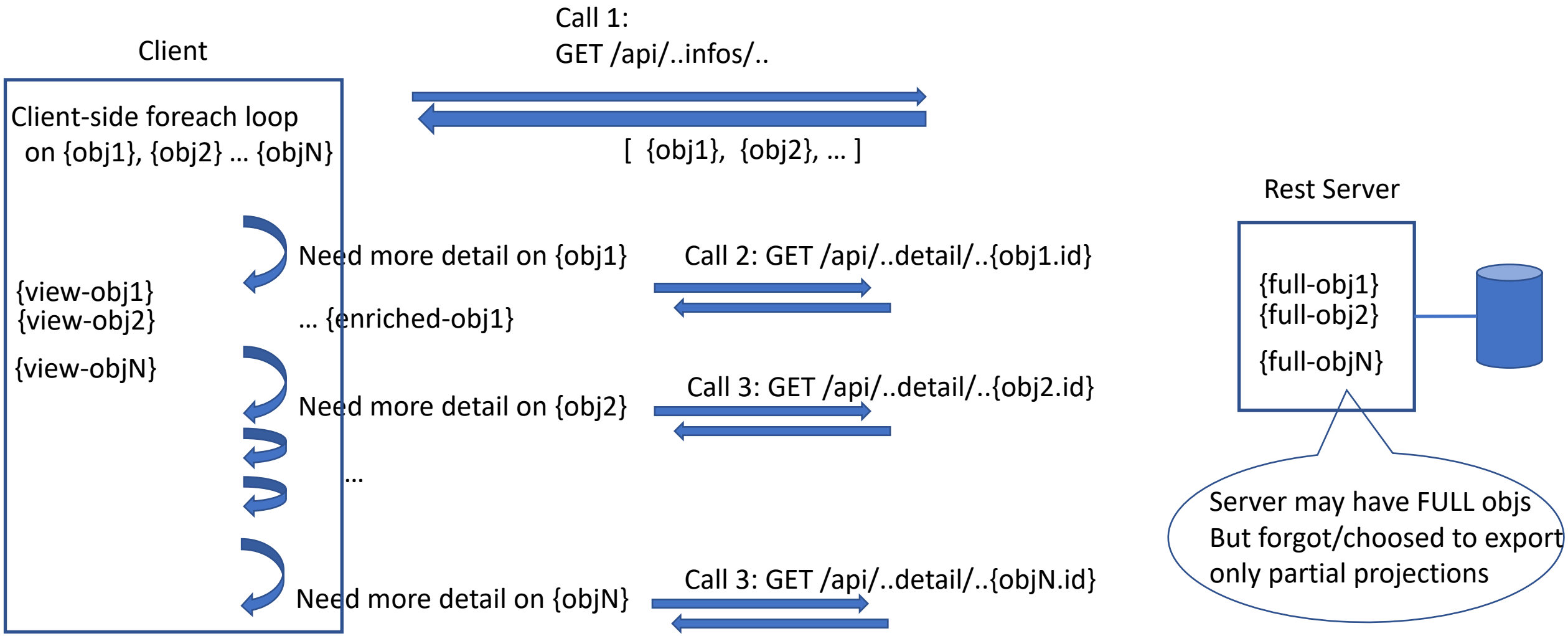
GraphQL to the rescue ..

Switch to 

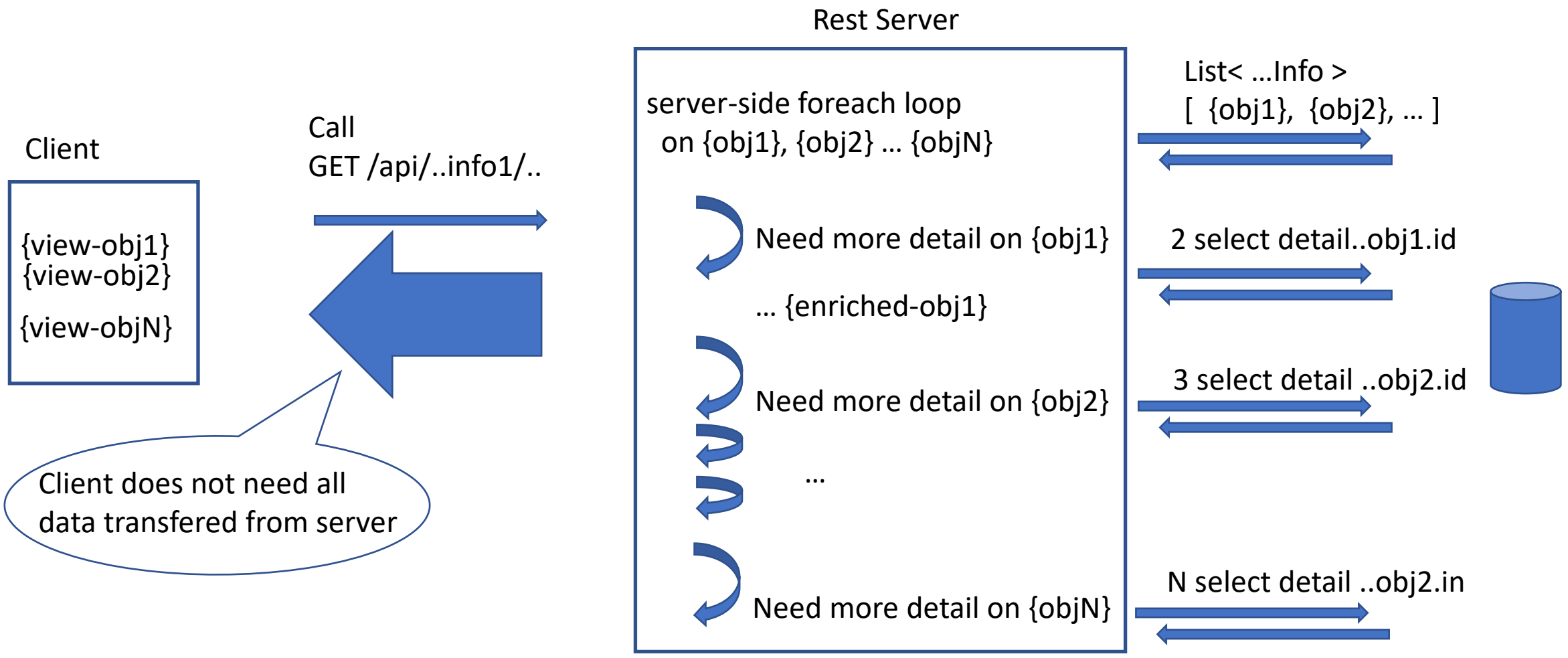
java + annotation + springboot ecosystem for GraphQL

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

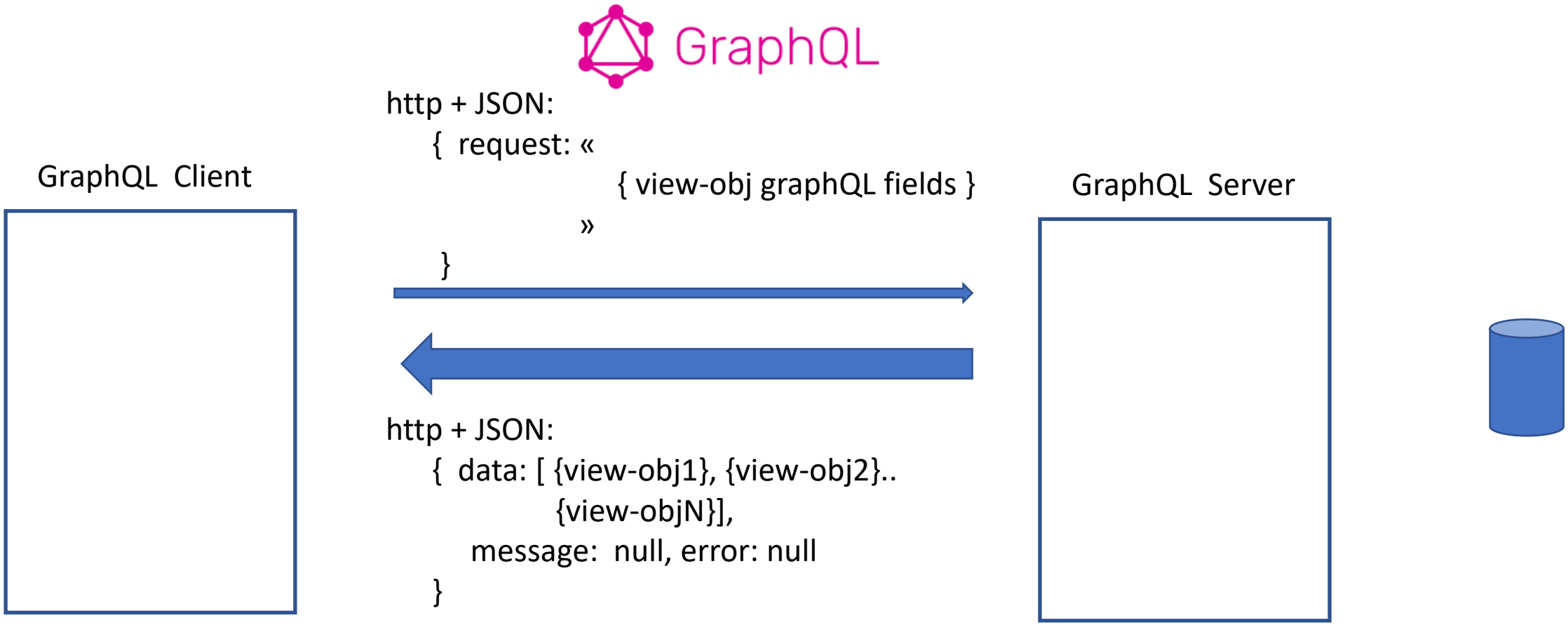
Direct JPA QL to DTO
(or even Native SQL to DTO)

```java
@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

```sql
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 dilemna


& 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