Programming 2.3

Web API - week 2

Karel de Grote Hogeschool

Practical Information

- Deadline for the first low-stake:
    - End of the third week?
    - End of spring break?
        - This means feedback will be posted later as well!

- GET
- DELETE
- POST
- PUT
- Model mapping
- Validation
- Serialization

GET endpoints

- **GET ALL:** `http://domain.tld/api/books`
  - All books - returns a list/array
  - Can include request parameters
- **GET 1:** `http://domain.tld/api/books/12`
  - Book with ID 12 - returns a single record/object
- **GET all ... of ...:** `http://domain.tld/api/books/5/authors`
  - All authors of the book with ID 5 - returns a list/array
  - Can include request parameters

An endpoint is the combination of the **verb** (or method) and the **path**, including path variables.

Request parameters should only filter by the (non-ID) properties of the resource being fetched.

Request parameters are *not* included in the samples that follow.

GET all

http://domain.tld/api/books

- Request

```
GET http://localhost:8080/api/books HTTP/1.1
Accept: application/json
```

- Response

```
HTTP/1.1 200
Content-Type: application/json

[
    {
        "id": 9,
        "title": "The Two Towers"
    },
    {
        "id": 8,
        "title": "The Fellowship of the Ring"
// etc.
```

This presentation only lists the minimum HTTP headers.

Multiple objects, so **[** and **]** at the top level.
JSON array, *even* if there's only one book. (API contract)

GET all

The codes marked with "X" must be taken care of inside the controller methods or using ControllerAdvice.

- Possible status codes:

Code	Description	Meaning	
200	OK	Record(s) found → message body	X
204	No Content	No records found → no message body	X
400	Bad Request	Invalid format for a request parameter	
404	Not Found	Path (resource) was not found	
405	Method Not Allowed	Endpoint exists, but not as GET	
500	Internal Server Error	Shouldn't happen (uncaught exception)	

401	Unauthorized	See "Spring Security" later.
403	Forbidden	

These status codes will not be discussed any further in this presentation. (see "Security")

GET 1

http://domain.tld/api/books/12

- Request

```
GET http://localhost:8080/api/books/12 HTTP/1.1
Accept: application/json
```

- Response

```
HTTP/1.1 200
Content-Type: application/json

{
    "id": 12,
    "title": "Black House"
}
```

A single object, so **{** and **}** at the top level.
JSON object.

GET 1

- Possible status codes:

Code	Description	Meaning	🍃
200	OK	Record was found → message body	X
400	Bad Request	Invalid path variable, not a number	
404	Not Found	Record with given ID was not found	X
405	Method Not Allowed	Endpoint (path) exists, but not as GET	
500	Internal Server Error	Shouldn’t happen (uncaught exception)	

GET all ... of ...

http://domain.tld/api/books/12/authors

- Request

```
GET http://localhost:8080/api/books/12/authors HTTP/1.1
Accept: application/json
```

- Response

```
HTTP/1.1 200
Content-Type: application/json

[
    {
        "id": 3,
        "name": "Stephen King",
        "dateOfBirth": "1947-09-21"
    },
    {
        "id": 4,
// etc.
```

JSON array, *even* if there's only 1 author! (API contract)

GET all … of …

- Possible status codes:

Code	Description	Meaning	
200	OK	Record(s) found → message body	X
204	No Content	No records found → no message body	X
400	Bad Request	Invalid format for a parameter (path or req.)	
404	Not Found	Record with given ID was not found	X
405	Method Not Allowed	Endpoint exists, but not as GET	
500	Internal Server Error	Shouldn’t happen (uncaught exception)	

There’s a difference between 204 and 404!

- GET
- DELETE
- POST
- PUT
- Model mapping
- Validation
- Serialization

DELETE 1

http://domain.tld/api/books/12

- Request

```
DELETE http://localhost:8080/api/books/12 HTTP/1.1
```

- Response

```
HTTP/1.1 204
Content-Type: application/json
```

DELETE 1

- Possible status codes:

Code	Description	Meaning	🍃
204	No Content	Record was found → no message body	X
400	Bad Request	Invalid path variable, not a number	
404	Not Found	Record with given ID was not found	X
405	Method Not Allowed	Endpoint (path) exists, but not as DELETE	
500	Internal Server Error	Shouldn’t happen (uncaught exception)	

- GET
- DELETE
- POST
- PUT
- Model mapping
- Validation
- Serialization

POST endpoints

- **POST:** `http://domain.tld/api/books`
    - Create a new book
- **POST nested:** `http://domain.tld/api/books/5/authors`
    - Add an existing author to an existing book

You may find different strategies and opinions on this matter! (i.e., “why not PUT on /books?”)
We’ll use this approach as a guideline.

POST

http://domain.tld/api/books

- Request

```
POST http://localhost:8080/api/books HTTP/1.1
Accept: application/json
Content-Type: application/json

{
     "title": "My First Book",
     "genre": "MYSTERY",
     "pages": 120
}
```

- Response

```
HTTP/1.1 201
Content-Type: application/json

{
     "id": 13,
     "title": "My First Book",
     "genre": "MYSTERY",
     "rating": null,
     "pages": 120
}
```

A single JSON object, twice. Properties are different! (at least the ID)

POST

- Possible status codes:

Code	Description	Meaning	
201	Created	Record was created → message body	X
400	Bad Request	Invalid record (properties, values, …)	
404	Not Found	Path (resource) was not found	
405	Method Not Allowed	Endpoint (path) exists, but not as POST	
500	Internal Server Error	Shouldn't happen (uncaught exception)	

If a "valid" record can't be created (i.e., unique constraint violation), then returning 400 is fine. Making a distinction with 409 is not needed for this course.

Validation framework and ControllerAdvice can be used to take care of the 400s.
No additional work in the controller methods.

POST nested

http://domain.tld/api/books/12/authors

- Request

```
POST http://localhost:8080/api/books/12/authors HTTP/1.1
Content-Type: application/json

{
    "id": 1
}
```

Adding author '1' to book '12'

- Response

```
HTTP/1.1 204
Content-Type: application/json
```

POST nested

- Possible status codes:

Code	Description	Meaning	🍃
204	No Content	Req. handled OK → no message body	X
400	Bad Request	Invalid record (properties, values, …)	
404	Not Found	Record with given ID was not found	X
405	Method Not Allowed	Endpoint (path) exists, but not as POST	
500	Internal Server Error	Shouldn’t happen (uncaught exception)	

POST

JS

```
fetch('/api/publishers', {
    method: 'POST',
    headers: {
        "Accept": "application/json",
        "Content-Type": "application/json"
    },
    body: JSON.stringify({
        name,
        yearFounded
    })
})
    .then(/* ... */)
    .then(/* ... */)
    .catch(/* ... */)
```

This object contains all of the request options.

Don't forget the headers! "Content-Type" *must* be quoted since a dash is not allowed for an identifier.

This object contains the payload (=HTTP message body).
This is shorthand notation for **`{ name: name, … }`**

- GET
- DELETE
- POST
- PUT
- Model mapping
- Validation
- Serialization

PUT

http://domain.tld/api/books/12

- Request

```
PUT http://localhost:8080/api/books/12 HTTP/1.1
Accept: application/json
Content-Type: application/json

{
    "id": 12,
    "title": "Updated title",
    "genre": "MYSTERY",
    "rating": null,
    "pages": 120
}
```

All fields should be included in a PUT request.
Full replace/overwrite.

- Response

```
HTTP/1.1 204
Content-Type: application/json
```

PUT

- Possible status codes:

Code	Description	Meaning	
201	Created	Record was created → message body	X*
204	No Content	Record was updated → no message body	**X**
400	Bad Request	Invalid record (properties, values, …)	
405	Method Not Allowed	Endpoint (path) exists, but not as PUT	
409	Conflict	Path ID doesn't match body ID	**X**
500	Internal Server Error	Shouldn't happen (uncaught exception)	

There's no need to implement "creation" for records that don't exist yet.

- GET
- DELETE
- POST
- PUT
- **Model mapping**
- **Validation**
- **Serialization**

Model mapping

- DTO = Data Transfer Object (REST API)
- VM = Viewmodel (MVC)
- Use an API for automatic mapping
  - Modelmapper (modelmapper.org)

```
implementation "org.modelmapper:modelmapper:3.0.0"
```

https://gitlab.com/kdg-ti/programming-2.3/assignments/cycling

Model mapping

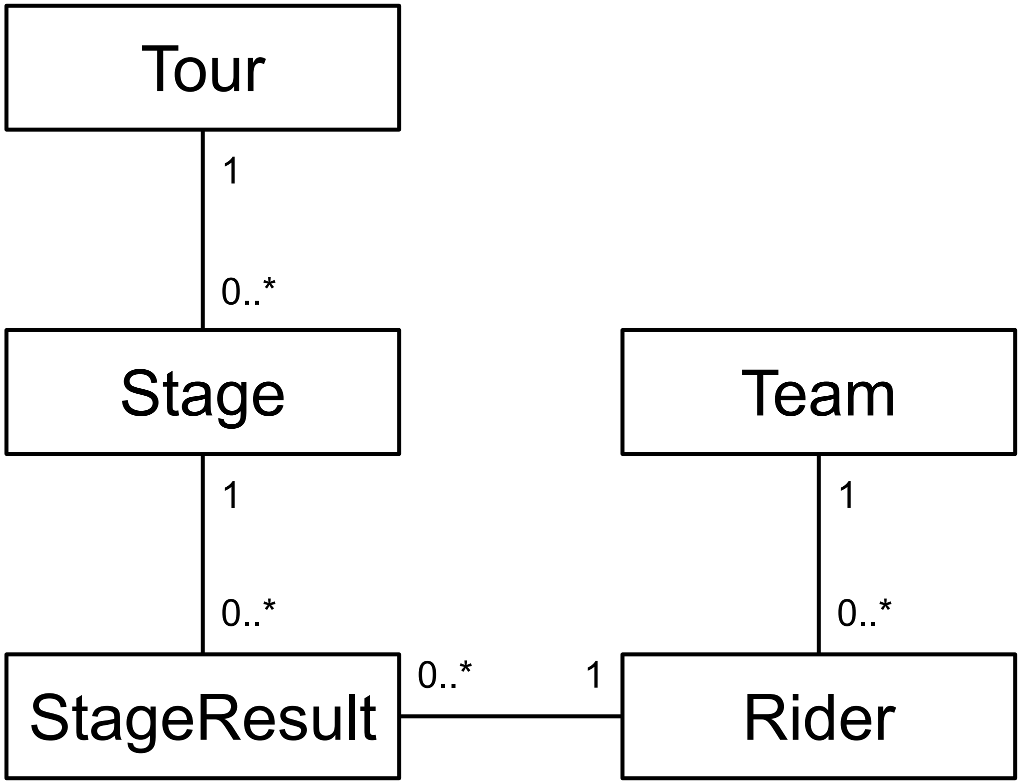
```
@Configuration
public class CyclingConfiguration {
   @Bean
   public ModelMapper modelMapper() {
       return new ModelMapper();
   }
}
```

```
var riderViewModel = modelMapper
                      .map(rider, RiderViewModel.class);
```

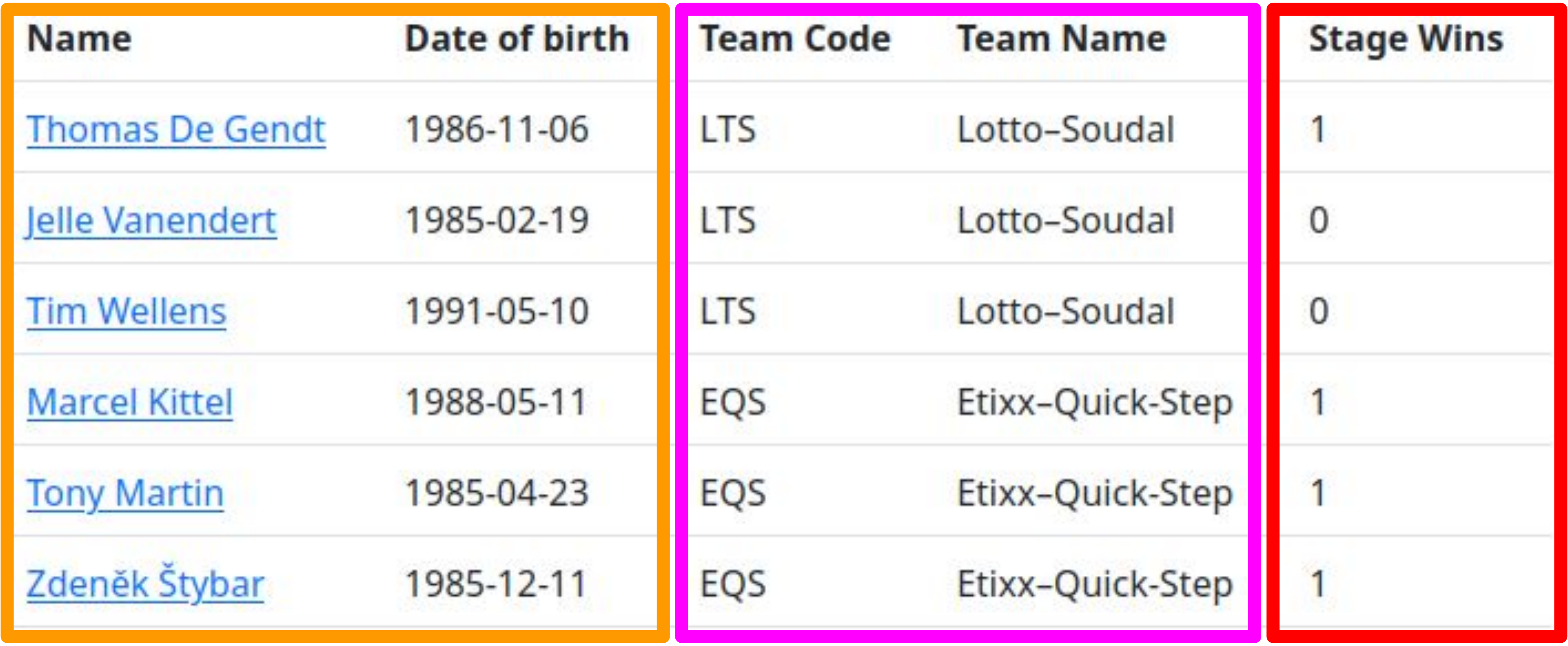
Consider **`modelMapper`** to be an “autowired” dependency.

VMs: example

- Domain:



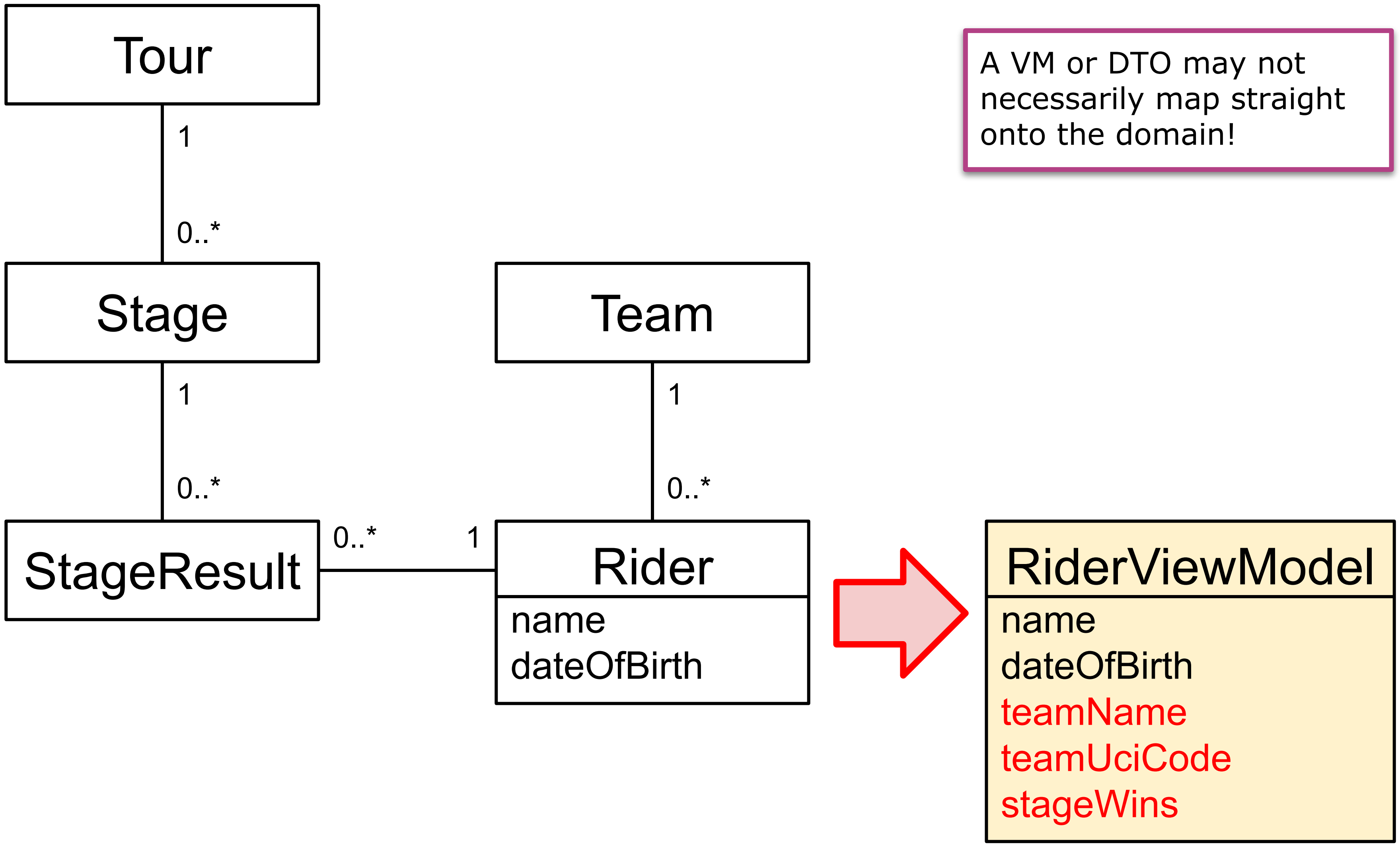
VMs: example



Name	Date of birth	Team Code	Team Name	Stage Wins
Thomas De Gendt	1986-11-06	LTS	Lotto–Soudal	1
Jelle Vanendert	1985-02-19	LTS	Lotto–Soudal	0
Tim Wellens	1991-05-10	LTS	Lotto–Soudal	0
Marcel Kittel	1988-05-11	EQS	Etixx–Quick-Step	1
Tony Martin	1985-04-23	EQS	Etixx–Quick-Step	1
Zdeněk Štybar	1985-12-11	EQS	Etixx–Quick-Step	1

Rider Team ???

VMs: example



A VM or DTO may not necessarily map straight onto the domain!

- GET
- DELETE
- POST
- PUT
- Model mapping
- Validation
- Serialization

Validation

- We'll continue to use Hibernate Validator
- We can use the annotations of the Bean Validation Framework:
  - Package: **`javax.validation`**
  - Add **`@Valid`** to the controller method's parameter(s)
  - Add validation annotations to the attributes of the VMs and DTOs:

    `@NotNull`, **`@Size`**, **`@Positive`**, ...

```
implementation "org.springframework.boot:spring-boot-starter-validation:2.6.3"
```

Validation

```
@PostMapping
public ResponseEntity<BookDto> createNewBook(
                     @RequestBody @Valid NewBookDto bookDto) {
    // ...


public class NewBookDto {
    @NotNull
    @Size(min = 3, max = 50)
    private String title;

    @NotNull
    private Genre genre;

    @Positive
    private int pages;

    // ...
```

- GET
- DELETE
- POST
- PUT
- Model mapping
- Validation
- **Content negotiation**

Content negotiation

- Let’s try XML …

- Request

```
GET http://localhost:8080/api/books
Accept: application/xml
```

- Response

```
HTTP/1.1 406
```

Not Acceptable 😲

Content negotiation

- By default, Spring Boot uses Jackson for (de)serialization
  - JSON support is included
  - XML support is *not* included

```
implementation "com.fasterxml.jackson.dataformat:jackson-dataformat-xml:2.13.1"
```

Content negotiation

- Let’s try XML again …
- Request

```
GET http://localhost:8080/api/books
Accept: application/xml
```

- Response

```
HTTP/1.1 200
Content-Type: application/xml;charset=UTF-8

<List>
    <item>
        <id>9</id>
        <title>The Two Towers</title>
    </item>
    <item>
         <id>8</id>
    // ...
```

Assignment: Cycling

- Check out the assignment description on Canvas

Project

- Check out the project description on Canvas

- **Important!** Your project is to be submitted **halfway** through **term 3** for the first Low-Stake assessment!