Programming 5

Web API - REST

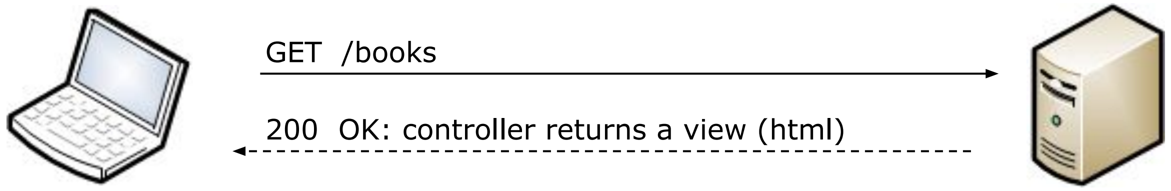
Karel de Grote Hogeschool

REST: Overview

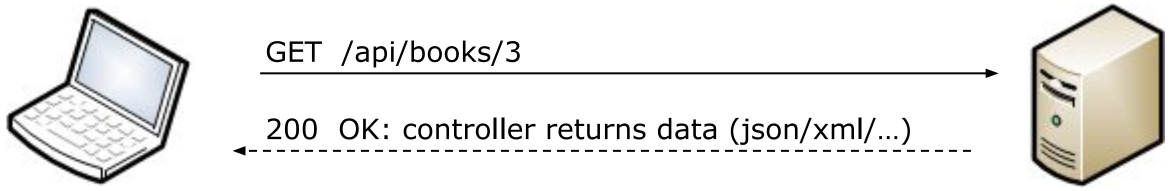
- **Web API and AJAX**
- **REST: Examples**
- **REST**
- **HTTP Header Fields**
    - **Content Negotiation**

MVC vs Web API

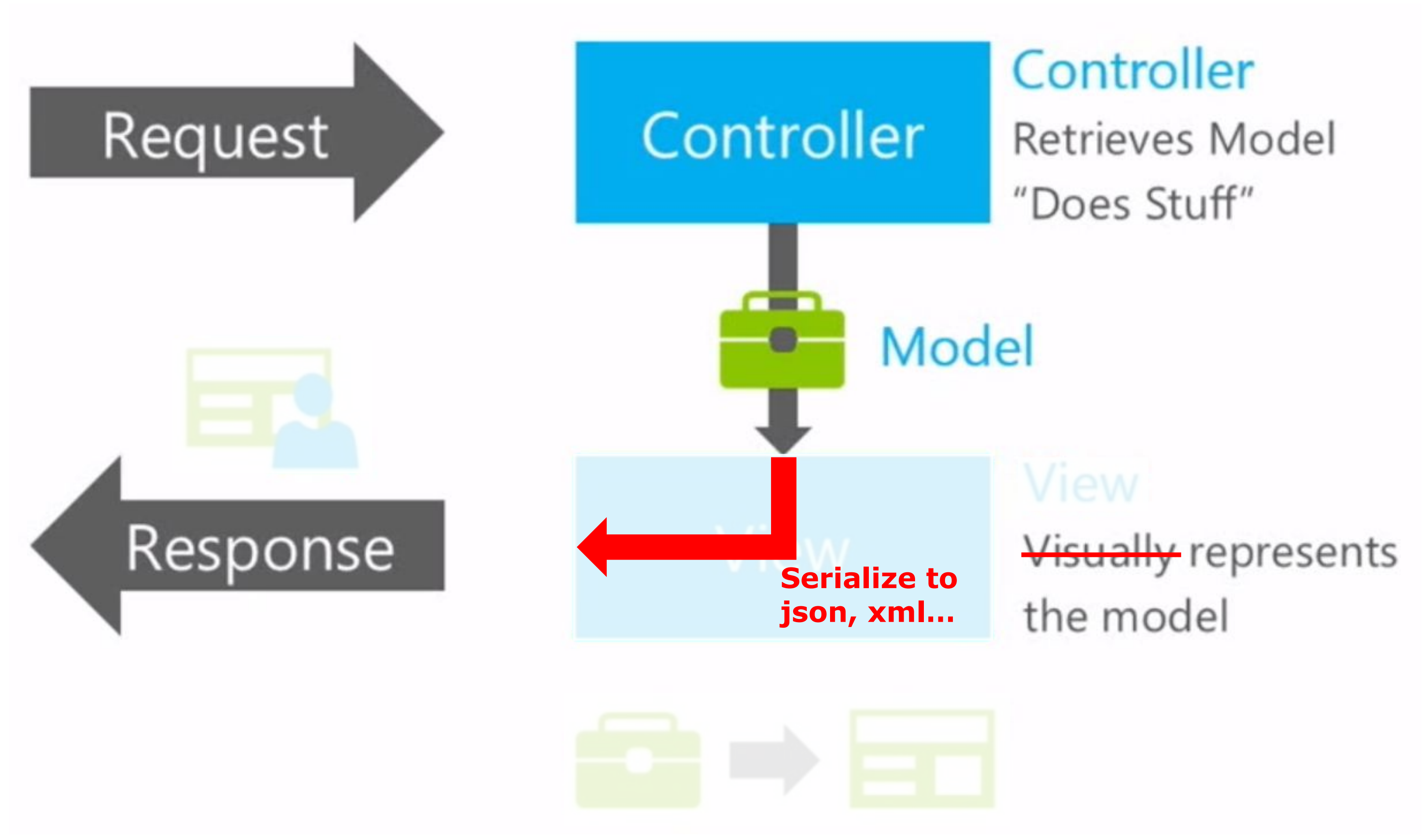
MVC: working with web pages



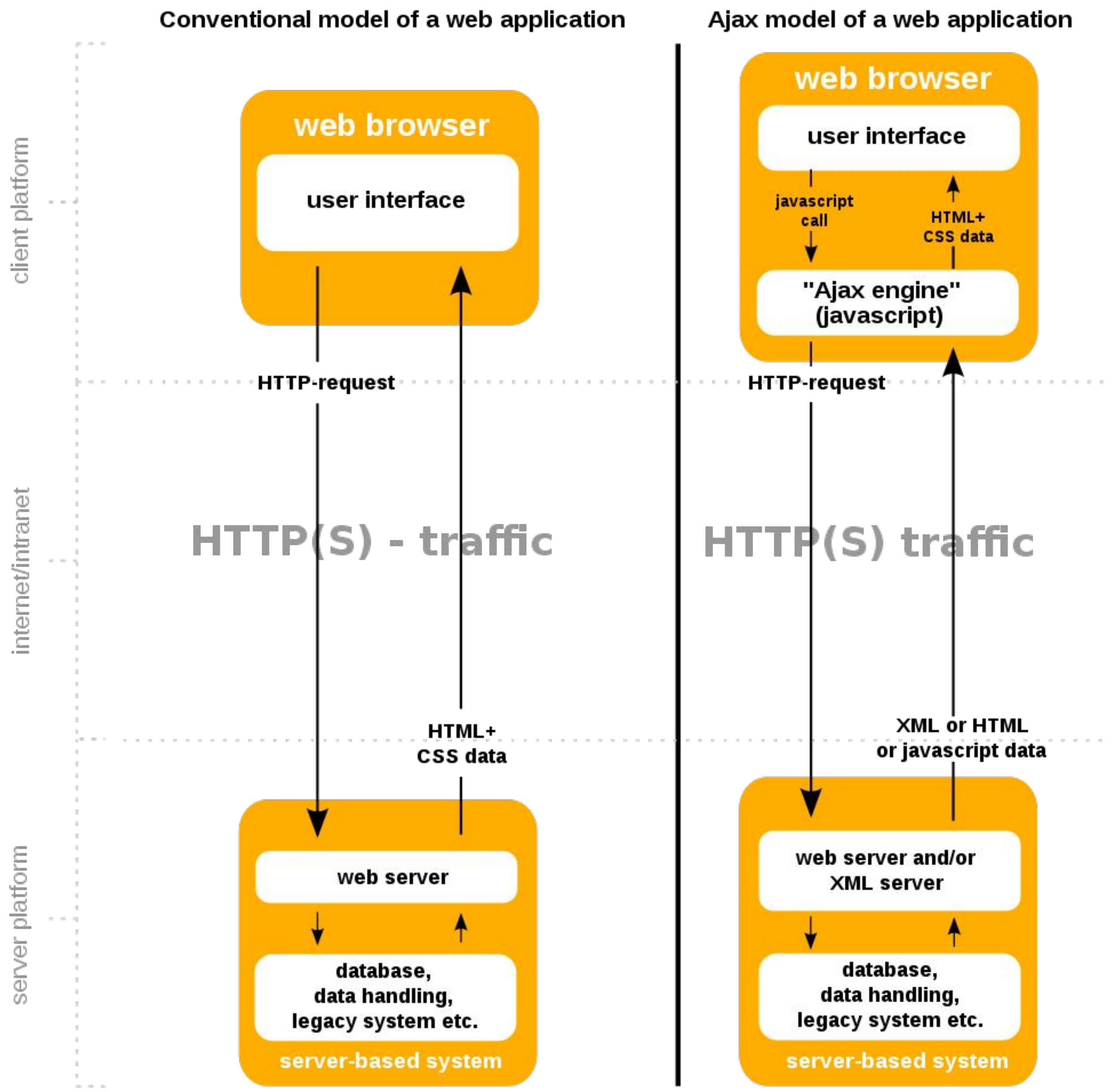
Web API: working with data



MVC vs Web API



AJAX


Conventional model of a web application
Ajax model of a web application
client platform
internet/intranet
server platform
web browser
user interface
HTTP-request
HTTP(S) - traffic
HTML+
CSS data
web server
database,
data handling,
legacy system etc.
server-based system
web browser
user interface
javascript
call
HTML+
CSS data
"Ajax engine"
(javascript)
HTTP-request
HTTP(S) traffic
XML or HTML
or javascript data
web server and/or
XML server
database,
data handling,
legacy system etc.
server-based system

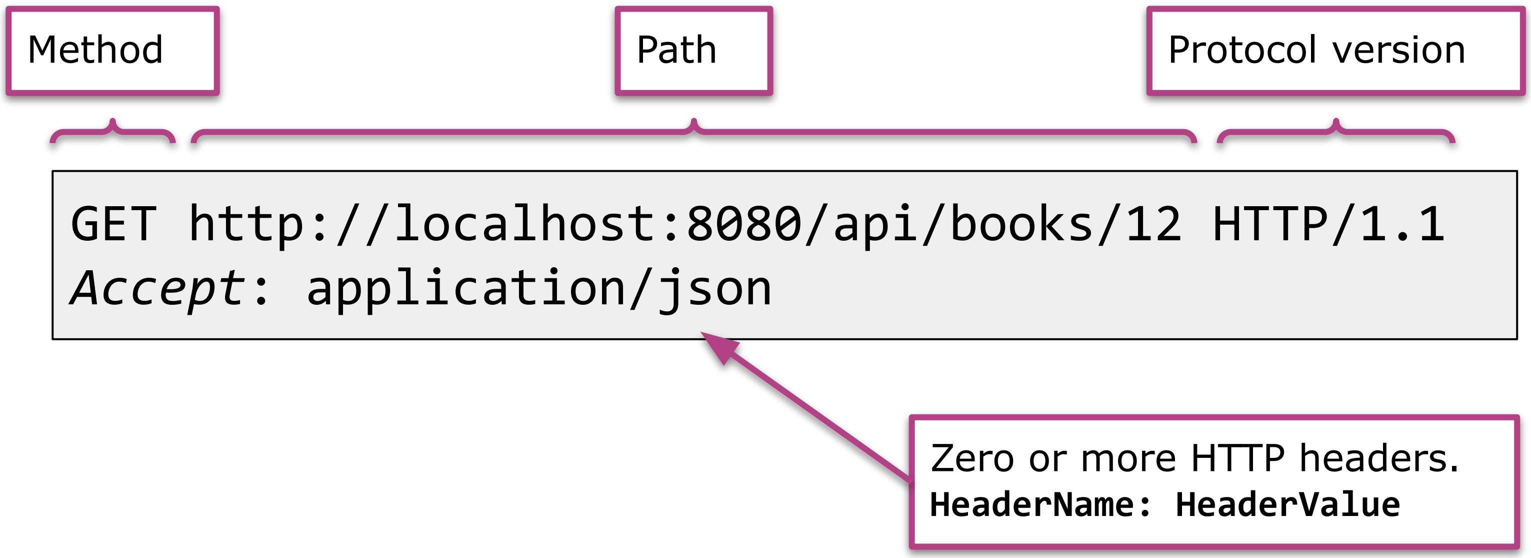
AJAX

- Asynchronous JavaScript and XML
- XML / JSON / …
- Technique enabled by web browsers (and, of course, the servers they're connecting to)

- Web API and AJAX
- REST: Examples
- REST
- HTTP Header Fields
  - Content Negotiation

REST - GET Example

- Request



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

- Additional data can be transmitted in the message body (see next couple of slides)

REST - GET Example

- Request

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

We "accept" a certain response...

- Response

```
HTTP/1.1 200
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 04 Feb 2022 08:36:11 GMT
Keep-Alive: timeout=60
Connection: keep-alive

{
    "id": 12,
    "title": "Black House",
    "genre": "HORROR",
    "rating": null,
    "pages": 700
}
```

Protocol version and status code

HTTP headers

Mandatory empty line

Message body

REST - POST Example

- 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
}
```

Mandatory empty line

Message body

- Response

```
HTTP/1.1 201
Content-Type: application/json
Transfer-Encoding: chunked
Date: Fri, 04 Feb 2022 09:06:25 GMT
Keep-Alive: timeout=60
Connection: keep-alive

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

This time, **`Content-Type`** is in *both* the request and the response

- Web API and AJAX
- REST: Examples
- REST
- HTTP Header Fields
  - Content Negotiation

REST

- **Re**presentational **S**tate **T**ransfer
  http://stackoverflow.com/questions/671118/what-exactly-is-restful-programming
- Architecture / set of conventions and best practices
- CRUD-actions
- Tied to the **HTTP** protocol

- Web API design best practices

REST - resources

- REST is 'resource-oriented' , not 'action-oriented'!
- The **url**(~uri) uniquely identifies a resource and not an action that is to be executed
- Resources within URIs are expressed in **plural**
- The action is determined by the HTTP **verb**

THANKS TO THE CLOUD, YOU CAN CONTINUE WORKING IN THERE, HOSKINS!
REST ROOM
© D.Fletcher for CloudTweaks.com

REST

- REST best practices are centered around:
  - The correct **verb**
    - GET, POST, PUT, PATCH, and DELETE
  - The correct **URL**
    - Including possible path and query params
  - The correct **message body**
  - The correct **status code**
  - The correct **header fields**
    - Content negotiation: `Accept` and `Content-Type`

REST - verbs

- Also called a ‘method’ (see HTML **`form`** tag)
- Indicates the kind of action

GET	Read operation
POST	Create operation
DELETE	Deletion / removal
PUT	Update (or create) a record in its entirety
PATCH	Partial update of a record (JSON merge patch RFC 7396 and JSON patch RFC 6902)

REST - verbs

Nouns	Verbs			
HTTP	**GET**	**POST**	**PUT**	**DELETE**
Collection i.e.: /books/	Retrieve a list of all elements	Create a new element		
Element i.e.: /books/123	Retrieve a specific element		Replace a specific element	Delete a specific element
CRUD	**Read**	**Create**	**Update**	**Remove**

+ Merge PATCH

REST - URLs

- Convention: prefix '**api**' and **lowercase**
  - `http://www.domain.tld/api/...`
- Examples:
  - `http://www.domain.tld/api/books`
    - All books
  - `http://www.domain.tld/api/books/5`
    - Book with ID 5
  - `http://www.domain.tld/api/books/5/authors`
    - All authors of the book with ID 5

REST - URLs

- Query parameters can be used as well
- Example:
  - `http://www.domain.tld/api/books?format=pdf`
- Try to avoid request parameters when filtering by related entities.

  For example: "*All authors of the book with ID 5*"
  - Use: `http://www.domain.tld/api/books/5/authors`
  - Not: ~~`http://www.domain.tld/api/authors?bookId=5`~~

REST - Parameters

- Query parameters
  - Used often with MVC
  - `http://www.domain.tld/api/books?format=pdf`

Its name is visible, even in the URL.

- Path variables
  - Used often with REST to identify a resource
  - `http://www.domain.tld/api/books/5`

- Both query parameters and path variables are **part of the URL**.
  - Used for filtering or identifying a resource
  - Actual records are sent with the **body**

REST - Status Codes

- HTTP status codes that we'll encounter and use:

200	OK	Success (*only* when the ones below are n/a)
201	Created	A new record has been created
204	No Content	Nothing to return (is different from Not Found!)
302	Found (Redirect)	Handled by Spring MVC (view: `"redirect: ... "`)
400	Bad Request	*Usually* handled by Spring (validation, …)
401	Unauthorized	Handled by Spring
403	Forbidden	Handled by Spring
404	Not Found	Record or page was not found
405	Method Not Allowed	Handled by Spring
409	Conflict	Inconsistency or incorrectness
500	Internal Server Error	Should not occur!

Summary - URLs, Parameters, and Body

		Path Variable	Query Params	Request Body
GET (all)	/api/books	NO	OPT	NO
GET (1 record)	/api/books/1234	MD	NO	NO
GET (all ... of ...)	/api/books/1234/authors	MD	OPT	NO
DELETE (1 record)	/api/books/1234	MD	NO	NO
POST	/api/books	NO	NO	MD
POST (Nested)	/api/books/1234/authors	MD	NO	MD
Merge PATCH	/api/books/1234	MD	NO	MD
PUT	/api/books/1234	MD	NO	MD

NO	Not Allowed
MD	Mandatory
OPT	Optional

Never use query variables to filter by ID!

Summary - Status Codes

	Ok 200	Created 201	No Content 204	Bad Request 400	Not Found 404	Conflict 409
GET (all)	X		X			
GET (1 record)	X				X	
GET (all ... of ...)	X		X		X	
DELETE (1 record)			X		X	
POST		X		X(*)		
POST (Nested)		X		X(*)	X	
Merge PATCH	-		X	X(*)	X	
PUT	-	-	X	X(*)	X	X

X	**Required**
X(*)	**@Valid**
-	**Optional**

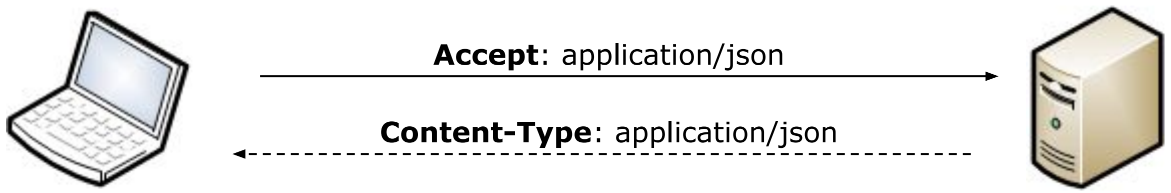
This is implemented easily by using a DTO with the @Valid annotation.

- Web API and AJAX
- REST: Examples
- REST
- **HTTP Header Fields**
  - **Content Negotiation**

Content negotiation

- HTTP request header
  - **`Accept`**
- HTTP response header
  - **`Content-Type`**

These are two headers that you must include whenever applicable.


Accept: application/json
Content-Type: application/json

Content negotiation

- Scenario 1: server doesn’t support XML
- Request

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

- Response

```
HTTP/1.1 406
```

Not Acceptable 😲

Content negotiation

- Scenario 2: server does support XML
- 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>
    // ...
```

Content negotiation

- Always use an `Accept` header when requesting information from the server which is to be returned in the body of the HTTP response
- Always use a `Content-Type` header when sending information to the server in the body of your HTTP request

	`Accept`	`Content-Type`
From client to server	Indicates the response type the client expects	Indicates the type of the request body the client sends
From server to client	N/A	Indicates the type of the response body the server sends

2

REST: All verbs

- **GET**
- **DELETE**
- **POST**
- **Merge PATCH**
- **PUT**

GET endpoints

- **GET ALL:** `http://domain.tld/api/books`
  - All books - returns a list/array
  - Can include query 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 query parameters

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

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

Query 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
- Merge PATCH
- PUT

DELETE 1

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

- Request

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

- Response

```
HTTP/1.1 204
```

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
- Merge PATCH
- PUT

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
}
```

Notice that there is no ID in the request body. Why? 🤔

- 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, …)	X*
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` should 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
```

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, …)	X*
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)	

- GET
- DELETE
- POST
- Merge PATCH
- PUT

Merge PATCH - without response

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

- Request

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

{
    "title": "Updated title"
}
```

Only include those fields that should be updated.

- Response

```
HTTP/1.1 204
```

Merge PATCH - with response

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

- Request

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

{
      "title": "Updated title"
}
```

Why is there an `Accept` header?

Only include those fields that should be updated.

- Response

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


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

Merge PATCH

- Possible status codes:

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

- GET
- DELETE
- POST
- Merge PATCH
- PUT

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
```

PUT

- Possible status codes:

Code	Description	Meaning	
200	OK	Record was updated → message body	-
201	Created	Record was created → message body	-
204	No Content	Record was updated → no message body	X
400	Bad Request	Invalid record (properties, values, …)	X*
404	Not Found	Record wasn’t found	X
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)	

- Optional