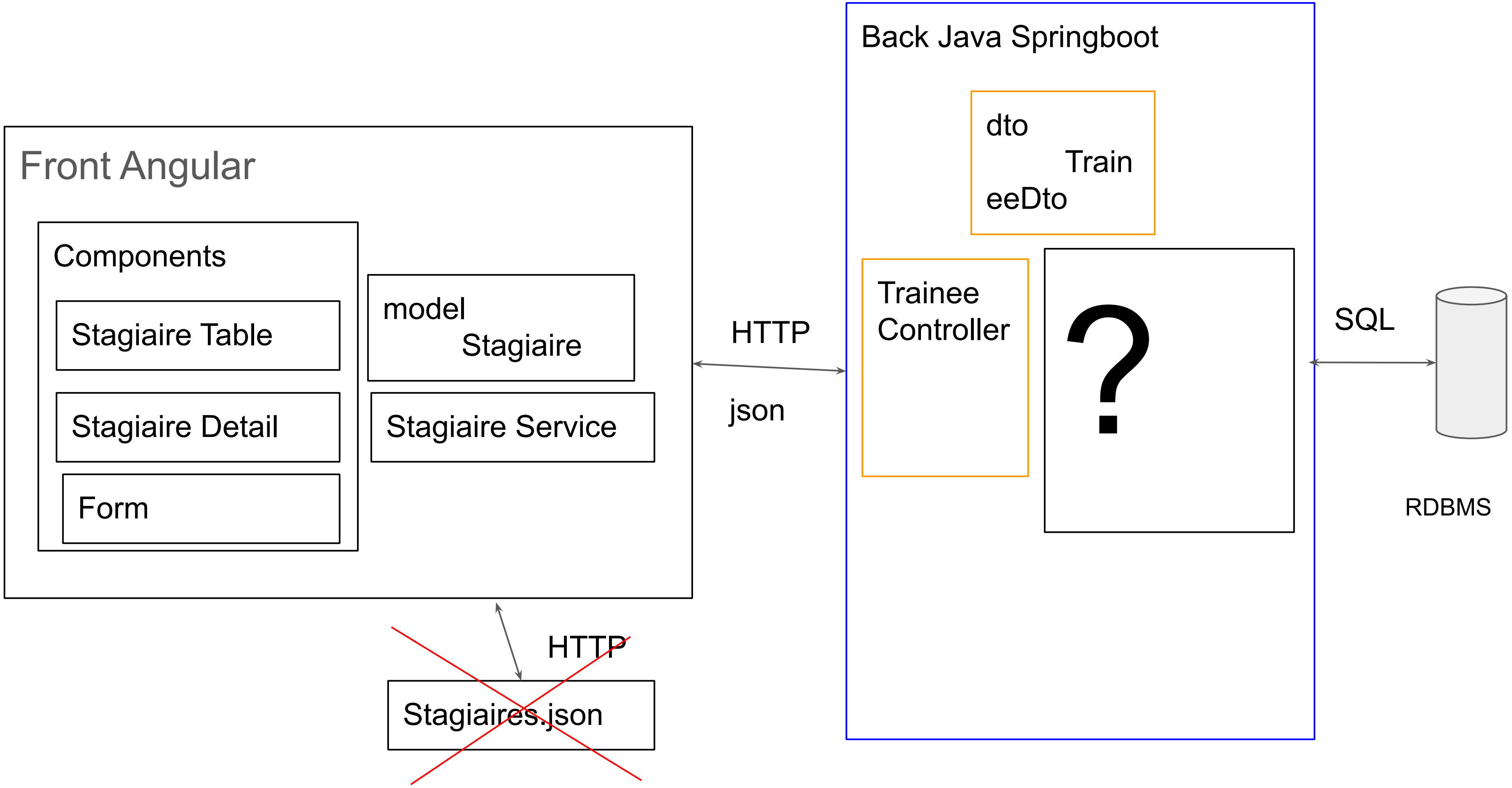
Backend Java with Spring(boot)


Front Angular
Components
Stagiaire Table
Stagiaire Detail
Form
model
Stagiaire
Stagiaire Service
HTTP
json
HTTP
Stagiaires.json
Back Java Springboot
dto
TraineeDto
Trainee
Controller
?
SQL
RDBMS

Spring

Framework Java with components to develop Backend Applications

A component Spring is often a wrapper to a JEE component

- Controller Web to deal with HTTP: Rest, MVC
- Spring Security: authentication, cryptography, CORS
- Spring Data: JPA, JDBC, MongoDB
- Reactive: asynchronous
- Tests
- Batch, Messaging

Spring

Framework 2 in 1

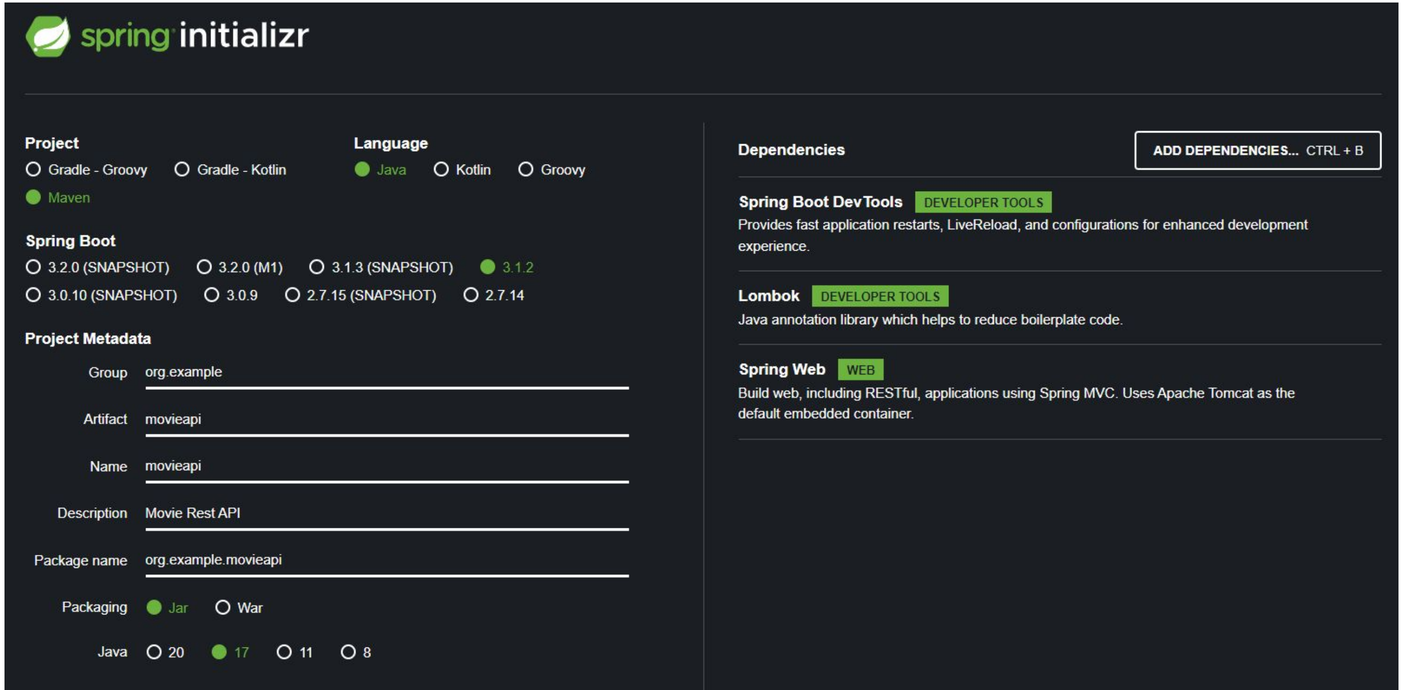
- Spring 6: Web Application to deploy in a Java Application Server (Tomcat, Wildfly)
- Spring Boot 3: standalone application containing
  - All dependencies: JEE, Spring, others
  - Application Server: Tomcat, Jetty

Spring and JEE

- Java EE 8 by Oracle: Spring 5, Spring Boot 2.7, Java **8-11**-17
  - javax.persistence.Entity (JPA)
  - javax.validation.constraints.NotNull (Bean Validation)
  - Tomcat 9
- Jakarta EE 9 (10) by Eclipse Foundation: Spring 6, Spring boot 3+, Java **17**
  - jakarta.persistence.Entity (JPA)
  - jakarta.validation.constraints.NotNull (Bean Validation)
  - Tomcat 10
- All Spring components are JEE Bean components
  - DI: Dependency Injection

Spring(boot)

- Main site: spring.io
- Spring Initializr: https://start.spring.io/



Project
Gradle - Groovy Gradle - Kotlin [✓ Maven]

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.2.0 (SNAPSHOT) 3.2.0 (M1) 3.1.3 (SNAPSHOT) [✓ 3.1.2]
3.0.10 (SNAPSHOT) 3.0.9 2.7.15 (SNAPSHOT) 2.7.14

Project Metadata

Group	org.example
Artifact	movieapi
Name	movieapi
Description	Movie Rest API
Package name	org.example.movieapi
Packaging	[✓ Jar] War
Java	20 [✓ 17] 11 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

HTTP(S)

- Protocole de communication Client-Server
- Domaines
  - Web: HTML + CSS + images + …
  - Web Service : SOAP, WSDL (XML) by W3C
  - API Rest : HTTP + JSON (ou XML)
- Elements
  - Headers
    - Method: GET, POST, PUT/PATCH, DELETE
    - Url: http[s]://www.example.org/api/movies
    - Status:
      - 1xx: Information
      - 2xx: OK
      - 3xx: redirection
      - 4xx: mauvaise demande du client
      - 5xx: erreur serveur
    - Other Headers: Accept, User-Agent, Content-Type, Content-Length, Date
  - Body: HTML, CSS, IMG, JSON, XML, …

HTTP(2)

- Headers for Request/Response
  - Content-Type: MIME-TYPE (image/png, text/html, application/json, …, */*)
  - Accept: MIME-TYPE(s)
  - Content-Length
  - Date
  - Server

Spring Boot Rest Controller

(De)Serialization default Content-Type:

- Simple data (String, int, double, boolean, …) : text/plain
- Object: application/json

For other media types: XML, CSV, … Spring just need a converter

Routing

2 ways

- A rest controller class (@RestController + @RequestMapping) with methods:
  - Entry point: method annotated with @GetMapping, @PostMapping, …
- A routing class : FunctionalRoute
  - Entry point: route => function

Example: controller running in tomcat listening at address localhost:8080

GET http://localhost:8080/api/movies/byTitle?t=Batman

```
@RestController @RequestMappint("/api/movies")
class  MovieController {
        @GetMapping("byTitle")
        Movie getByTitle(String title)
}
```

Classic Routing

Resource: /api/movies

method	path	semantic
GET		Read All Movies
GET	123	Read Movie #123
POST		Add new Movie
PUT/PATCH	123	Update Movie #123
DELETE	123	Delete Movie #123

Rest controller: params

- query param: url?t=Batman&y=2022
  - 1st param: name=t value=Batman
  - 2nd param: name=y value=2022
  - @RequestParam with validation:
    - Required
    - Conversion String => boolean, int, double, …, LocalDate, …
- path param: /api/movies/{id}
  - Example: /api/movie/123
  - @PathVariable
- body param: complex data (Collection, Custom Object)
  - @BodyRequest

Rest Controller: Custom Data

- Java class with default constructor and getters/setters
- Java 17 immutable Records
- Converter JSON: Jackson

Rest Controller Development

1. Define a service interface: MovieService
2. Write Unit Tests for all methods of controller: MovieController
3. Correct code of the methods tested until success

DI: Dependency Injection

What can be injected: @Bean (java/jakarta EE)

Spring beans:

- @Component
- @Controller, @RestController
- @Service
- @Repository

Spring Data

https://spring.io/projects/spring-data

- JPA
- JDBC
- MongoDB
- others …

Spring JPA Repository

- CRUD
  - save, saveAll, saveAndFlush, saveAllAndFlush
  - (update)
  - delete, deleteById, …
  - findById
  - findAll
  - findAll(Sort)
  - findAll(Pageable)
  - findAll(Example)
- Add business methods
  - with method name only: SQL query automatically generated
  - with JPA JPQL query
  - with JPA entity graph
  - with native SQL (vendor dependant)