Leerdoelen Spring-fase

versie 1.0.1, 20231107

- Een applicatie opzetten met Spring Initialzr
    - project types: Maven, Gradle Groovy, Gradle Kotlin
        - maven en gradle wrapper: doel en dat je ze in version control zet
    - wat *is* een dependency?
    - Ken typische dependencies
        - Spring Data/JPA
        - Spring Web
        - PostgreSQL driver
        - Lombok
        - Devtools
- Controllers maken
    - wat is een API?
    - wat is JSON?
    - wat is XML?
    - wat is HTTP en hoe werkt het?
        - request (met verb)
        - response (met status code)
        - headers
        - body
    - wat is CRUD?
    - wat is REST?
        - GET (contract + implementatie)
        - POST (contract + implementatie)
            - 201 Created +
        - DELETE (contract + implementatie)
        - PUT (contract + implementatie)
        - PATCH (contract + implementatie)
    - Spring annotations for controllers:
        - @RestController
        - @RequestMapping
        - @GetMapping
        - @PostMapping
        - @VERBMapping (delete, put, patch, etc)
        - @PathVariable
        - @RequestBody
        - @RequestHeader
        - @RequestParam
        - @CrossOrigin
    - ResponseEntity
- Koppelen met database
    - wat is een database?
    - wat is een DBMS?
    - wat is een tabel?

    - wat is een rij
    - wat is een kolom?
    - wat is een primary key?
    - wat is een natural key?
    - wat is een surrogate key?
    - wat is een foreign key?
    - installeren van PostgreSQL
    - koppelen van database aan Spring (application.properties)
    - cascading: wat is het?
    - wat is de object-relational mismatch?
        - granulariteit
        - inheritance
        - identiteit
        - linking (als in tabel)
        - lazy vs eager loading en het n + 1 -probleem

- Hoe werkt Spring?
    - De container
    - Dependency Injection
    - Inversion of Control
    - @Autowired (en alternatieven)
    - @Bean
    - @Component
    - @Configuration
    - @ComponentScan
    - stereotype annotations
- databases in Java: termen
    - JDBC
    - JPA
    - Hibernate
- databases in Spring: de tabel
    - @Entity
    - @Id
    - @GeneratedValue
    - mutipliciteiten:
        - @OneToOne
        - @OneToMany
            - mappedBy
        - @ManyToOne
        - @ManyToMany
    - ? @JoinColumn
    - 'Pseudoniemen'
        - @Table
        - @Column
    -
- database in Spring: het systeem
    - Entity

    - Repository
        - CrudRepository
        - JpaRepository
        - PagingAndSortingRepository
    - @RestController
    - Seeding
        - data.sql en schema.sql
        - CommandLineRunner
    - Service (?)
    - Paging
    - Sorting
    - Queries into methods
    - De repository: hoe het echt werkt
    - Wat betekent "transient"
    - Optional
- Geavanceerd getten
    - Optional
    - eager vs lazy loading
    - communicatie met frontend
        - "natuurlijk"
        - @JsonIgnore, @JsonBackReference, @JsonManagedReference
        - DTOs (en records)
- Lombok
    - @Getter
    - @Setter
    - @NoArgsConstructor
    - @AllArgsConstructor
- Security
    - CORS
    - XSS
    - CSRF
    - OAuth2
    - authenticatie
    - authorisatie
    - PasswordEncryptie
    - Roles en RBAC
    - UserDetailsService
    - Filters

"Kan-doelen"

1. Maak een Spring Web app zonder database, met alleen de browser (en een ArrayList ofzo als 'database')
    - Spring Web
    - @RestController
    - Theorie: HTTP: requests en responses, verbs, preview: status codes, headers, body
    - @GetMapping

    - Theorie: waarom 8080 (en wat is TomCat?)
    - @RequestMapping
    - @PathVariable (voor een GetById)
2. Breid dat uit met Postmapping (wel HTTPie of Postman nodig)
    - @PostMapping
    - @RequestBody
    - Hoe je de 201-created-maakt
        - ResponseEntity
        - UriComponentsBuilder
        - Ehm... waar komt die UriComponentsBuilder vandaan? IoC...
    - @DeleteMapping met NoContent
3. PUT en PATCH
    - Theorie: weet het verschil tussen PUT en PATCH
    - PUT-contract ook 204
4. Ga nu een database toevoegen:
    - Spring Data/JPA en PostgreSQL driver
    - application.properties
    - database maken in pgAdmin (noot: databasenaam is hoofdlettergevoelig)
    - @Entity
    - @Id
    - @GeneratedValue
    - `JpaRepository<Item, Long>`
    - @Autowired (anders null)
    - Standaardmethoden repos (findAll, findById, save)
    - Querymethoden maken (findByName)
    - Merk mogelijk de noodzaak van een no-args-constructor
    - Merk mogelijk de noodzaak van getters...
5. Ga nu de database seeden, en grotere requests doen
    - CommandLineRunner
    - Probeer Spring Data REST uit
    - Theorie: waarom paging en sorting
    - Implementeer paging en sorting
6. Relaties opzetten
    - Bijvoorbeeld recept-ingredient-recept-ingredient
    - maak meerdere entities
    - discussie/demo: 'horizontale' versus 'vertikale' packages
    - @OneToMany (met mappedBy), @ManyToOne
    - Collecties moeten interfaces zijn! (meestal Set)
    - tags (ZOET, KINDEREN) @ManyToMany
7. Complexere data inlezen en opslaan
    - record, DTO
    - wat betekent "transient"?
8. Complexere data teruggeven
    - ken de niveaus
        - simpel: niets nodig
        - half-simpel: JsonIgnore OF JsonManagedReference/JsonBackReference
        - complex: maak je eigen DTO (bv met een record)
9. Lol met Lombok
    - probeer code te vervangen met @Setter, @Getter, @NoArgsConstructor...

10. Introducie services
    - voordelen/nadelen van services
    - service aanmaken voor een controller
11. @Transactional: waarvoor dient dat?
12. Security : mogelijk beter als er ook een front-end is?

Niet noodzakelijk voor eindopdracht maar wel zeer handig voor je carriere

1. Leer SQL met bv W3Schools of CodeAcademy, probeer data.sql en schema.sql
2. Bestudeer Spring REST, die controller-methodes automatisch aanmaakt