Techno Web II

DUT Informatique 2e année – S4

Quentin LEULY

I. Objectifs du cours

- Compréhension des principes de REST avec HTTP
- Introduction à Spring avec Spring-boot (injection de dépendance, web, JPA)
- Programmation d'un API REST avec Spring-boot

II. Resources utiles pour le cours

- Le cours sur Spring est disponible à l'adresse suivante : https://github.com/Kaway/monpetitbonsai/blob/main/spring_course/introduction_spring_boot_leuly_quentin.pdf
- Les spécifications de l'API à créer sont disponible à l'adresse suivante : https://github.com/Kaway/monpetitbonsai/blob/main/README.md
- Un schéma de l'architecture du project est disponible à l'adresse suivante : https://github.com/Kaway/monpetitbonsai/blob/main/schemas/mpp_architecture.drawio.png
- IDE conseillé : Intellij IDEA Community Edition (ou la version pour étudiants si vous avez votre propre PC)

III. Modalité du cours

- Travail par groupe de 2 ou 3 **maximum**
- Utiliser un gestionnaire de version (Git sur Github ou Gitlab) pour ne pas perdre votre code
- L'évaluation sera faite le dernier après-midi du module (20mn par groupe)

IV. Travail à faire

1. Sujet

Le but de l'exercice est de créer le back-end d'une application web. Ce back-end pourrait par exemple être utilisé par une application mobile ou une application front en JavaScript (ReactJS ou Angular pour ne citer que les plus utilisés).

Vous êtes chargés de développer une API de gestion de bonsaïs. A partir de cette API, il sera possible de créer, lire, modifier, supprimer des bonsais (CRUD[1]). Les bonsais auront (ou non) un propriétaire. Vous devrez aussi développer le CRUD pour les propriétaires des bonsais.

2. Initialisation du projet

- Cloner le projet de base à l'adresse suivante : https://github.com/Kaway/monpetitbonsai-dut et l'ouvrir avec votre IDE
- Récupérer les dépendances avec Maven
- Lancer l'application et depuis votre navigateur aller à l'adresse : http://localhost:8080/swagger-ui/
- Créer une base de données sous PostgreSQL (modifier le fichier « *resources/application.properties* » pour le faire correspondre à vos paramètres)
- Relancer l'application. Il ne doit plus y avoir d'erreurs au démarrage

3. Développement des fonctionnalités

Prenez le temps de lire les spécifications à l'adresse suivante : https://github.com/Kaway/monpetitbonsai.

Dans la base du projet que je vous ai donnée, certains éléments sont manquants (méthodes, attributs de classes, classes …). Vous devez les identifier et les ajouter.

1 Create, Read, Update, Delete

a) Bonsai

Pour le bonsai, vous ne traiterez pas les attributs *last_watering*, *last_repotting*, *last_pruning*. Le champ *owner_id* sera traité dans la partie Owner.

Commencez par créer la table « *bonsai* » de votre base de données. Vous identifierez les champs nécessaire grâce aux spécifications de l'API sur Github (https://github.com/Kaway/monpetitbonsai#operations-on-bonsais).

- Récupérer la liste de tous les bonsais (https://github.com/Kaway/monpetitbonsai#retrieve-all-bonsais-data). Vous ne traiterez que le paramètre « *status* » (?status=XXXX), permettant de ne récupérer que les bonsais dont le statut est égal au paramètre.
- Récupérer un bonsai (https://github.com/Kaway/monpetitbonsai#retrieve-data-on-a-specific-bonsai)
- Créer un bonsai (pas de lien avec le Owner pour l'instant) (https://github.com/Kaway/monpetitbonsai#create-a-bonsai)
- Supprimer un bonsai (https://github.com/Kaway/monpetitbonsai#modify-information-about-a-bonsai)
- Modifier un bonsai (https://github.com/Kaway/monpetitbonsai#modify-information-about-a-bonsai)
- Modifier le statut d'un bonsai (https://github.com/Kaway/monpetitbonsai#modify-information-about-a-bonsai)

b) Owner

Créez la table « *owner* », en n'oubliant pas le lien avec la table « *bonsai* » (un owner peut avoir plusieurs bonsais).

Créer deux nouvelles classes, BonsaiDTO et Bonsai, dans le package « *owner* ». Vous identifierez les champs nécessaire en analysant les spécifications de la partie Owner (https://github.com/Kaway/monpetitbonsai#operations-on-owners). **On ne réutilisera pas les classes du package « *bonsai* » dans ce package.** Cela nous permet d'avoir une modélisation correspondante à chaque point de vue (les *entities* restent par contre partagées, car elles représentent le lien avec la base de données, qui elle ne change pas).

- Récupérer la liste de tous les owners (https://github.com/Kaway/monpetitbonsai#get-bonsais-owners-list)
- Récupérer un owner (https://github.com/Kaway/monpetitbonsai#retrieve-data-on-a-specific-owner)
- Créer un owner (https://github.com/Kaway/monpetitbonsai#create-an-owner)
- Récupérer la liste des bonsais d'un owner (https://github.com/Kaway/monpetitbonsai#retrieve-bonsais-list-of-a-specific-owner)
- Transférer un bonsai d'un owner à un autre (https://github.com/Kaway/monpetitbonsai#transfer-a-bonsai-to-an-owner)
- Ajouter un bonsai à un owner (https://github.com/Kaway/monpetitbonsai#add-a-bonsai-to-an-owner)

c) CareEvents (hors documentation Github)

Les « *CareEvents* » sont les actions qu'un propriétaire peut effectuer sur un Bonsai. Il y a 3 types d'action : arrosage (« *watering* »), taille (« *pruning* »), rempotage (« *repotting* »).

Un CareEvent est constitué d'un datetime, de l'id du bonsai, du type d'event, et de l'id du owner ayant fait l'action (dans un souci d'historisation). L'unicité des données sera assurée par le tuple *{datetime, id bonsai, event type}*. Une clé primaire (UUID) sera tout de même générée pour cette table.

- Créer la table *care_event* d'après les spécifications données au-dessus.
- Lier l'entity *CareEventEntity* à l'entity *OwnerEntity*
- Créer la route ***GET /bonsais/{id}/care-events*** permettant de récupérer la liste de tous les évènements d'un bonsaï
- Créer la route ***POST /bonsais/{id}***, qui prendra comme body un DTO permettant de créer un évènement sur un bonsaï
- Sur la route ***GET /bonsais/{id}/care-events***, rajouter un paramètre (***?type=***) permettant de ne récupérer qu'un seul type de CareEvent
- Créer la route ***DELETE /bonsais/{id}/care-events/{event-id}*** permettant de supprimer un évènement

V. Le cours en hyper condensé

- Spring est un framework permettant de mettre en œuvre l'**injection de dépendance** ; c'est-à-dire que le framework peut gérer à votre place le cycle de vie de vos objets (vous n'avez pas besoin d'instancier vos services, Spring le fait pour vous).
- On distingue 3 grands types de **Beans** que Spring peut gérer pour vous : Les controllers (gestion des requêtes HTTP), les services (logique application), et les repositories (requêtes à la base de données).
- Vous pouvez déclarer un service avec **@Service**.
- Pour créer un controller web sous Spring-boot, on ajoute à une classe les annotations **@RestController** et **@RequestMapping**.
- Les repositories, quant à eux, sont annotés avec **@Repository**.
- Pour les autres classes dont vous souhaiteriez laisser la gestion à Spring, on pourra utiliser l'annotation **@Component**.
- Pour déclarer une méthode d'un controller comme pouvant traiter des requêtes web, l'annotation avec **@GetMapping**, **@PostMapping**, **@PatchMapping**, **@PutMapping** ou **@DeleteMapping**.
- Une Entity est une représentation d'une table de votre base de données. Les entities sont annotées par **@Entity** et **@Table**. Un de leurs champs doit être obligatoirement annoté par **@Id**, afin d'indiquer à JPA la clé primaire.