Framework – Initiation à Spring avec Spring-boot

CSID 2020 – 2021 · TP 1

Vendredi 13 novembre 2020

I. Initialiser le projet

IDE recommandé : IntelliJ Idea Community Edition

Recommandé : Installation de Maven (mais IntelliJ Idea embarque un maven), Java 11 minimum

Cloner le projet disponible à l'adresse suivante : https://github.com/Kaway/CSID-web-repository-base

II. Juste un controller

Créer une classe RepositoryController. Cette classe sera annotée par *@RestController* et *@RequestMapping("/repositories")*

- Développer un endpoint GET avec comme path "/repositories" permettant de récupérer une liste de repositories. Cette liste de répositories sera une variable initialisée dans le contructeur de notre controller
- Développer un GET endpoint sur /repositories/{name} qui renvoie l'objet Repository de la liste avec {name} comme nom
- Développer un POST sur /repositories qui prend en body le json d'un objet Repository. La méthode devra générer un UUID, modifier l'objet json du body pour y mettre l'UUID, afficher l'objet en console et retourner un HTTP 201 (created) avec comme valeur pour le *header Location* le nom du repository créé du repository
- Développer un DELETE et un PATCH et un PUT qui vont respectivement supprimer, mettre à jour partiellement et mettre à jour complètement les objets Repository contenu dans la liste de votre controller

III. Un service maintenant

Créer une classe RepositoryService. Cette classe sera annotée par @Service.

- Ajouter un contructeur au RepositoryController, qui prendra comme paramètre notre service. Annoter la méthode avec @Autowired (l'annotation est optionnelle, mais explicite)
- Déplacer la logique des méthodes de votre controller vers votre classe Service. Votre controller doit maintenant appeler votre service.
- Créer une classe RepositoryDTO sur le même modèle que Repository. Les méthodes de votre controller recevront dorénavant des RepositoryDTO, les transformeront en Repository avant d'appeler le RepositoryService

IV. La base de données, finalement

Créer une classe RepositoryRepository (oui, oui …) et annotez là avec *@Component*. Cette classe service à appeler la base de données à travers un JPARepository (oui, encore un repository) et à appeler Github via son API REST. Ensuite, on aggrègera les données afin de les transformer en objet métier.

- Créer les scripts nécessaire à la création d'une table "repository". Un repository est constitué d'un nom, d'un propriétaire, d'un nombre de "issues" et d'un nombre de "pull-requests". Le nom du repository sera unique
- Créer une classe RepositoryEntity, qui sera annotée par *@Entity*. Cette classe sera la représentation de notre base de données. Ajouter l'annotation *@Table(…)* afin que cette entity pointe vers la bonne table de la base de données
- Ajouter à cette classe les attributs nécessaires afin de pouvoir récupérer les éléments en base de données : les éléments seront annotées par *@Column*, *@Id*
- Créer une classe RepositoryDao qui sera annotée par *@Repository* et implentera l'interface JpaRepository<T, ID> de spring-jpa, où T représente le type de l'objet que l'on manipule (ici un "RepositoryEntity" et ID le type de la clé primaire
- Assemblez le tout !
- Dans la classe RepositoryRepository, vous injecterez le RepositoryDao, et en vous basant sur les méthodes de ce dernier, vous créerez les méthodes pour :
  - récupérer tous les repositories
  - récupérer un repository par son nom
  - supprimer un repository par son nom
  - mettre à jour un repository
  - créer un un repository