Framework – Initiation à Spring avec Spring-boot

CSID 2020 – 2021 · TP 2

Vendredi 27 novembre 2020

I. Initialisation

- Créer 2 répositories de tests sur Github (si vous n'avez pas encore de compte ou ne souhaitez pas utiliser votre compte personnel, créer un compte). Initialiser les repositories avec un **README.md**
- Ajouter quelques issues à ces repositories et ajouter quelques pull-requests (créer une nouvelle branche, faire une modification sur le **README.md**, commit, puis créer la pull-request)
- A partir des enpoints développés dans le TP 1, enregistrer les informations de ces 2 repositories dans votre bases de données

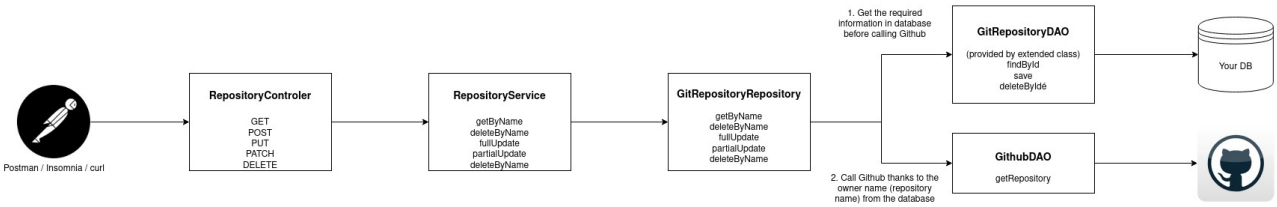
II. Le RestTemplate

- Créer une classe GitRepositoryRepository ( ). Annotez cette classe avec **@Repository** (ou **@Component**). Cette classe va maintenant servir d'intermédiaire entre notre RepositoryService et notre GitRepositoryDao: faites le nécessaire pour que le GitRepositoryService n'appelle plus GitRepositoryDao mais notre nouvelle classe.
- Créer une classe GithubRepositoryDao. Faites en en bean managé par Spring (utiliser la bonne annotation).
- Injecter un RestTemplate à cette classe. Le RestTemplate est un objet qui va nous permettre de faire des appels HTTP vers une API REST. Il offre des méthodes permettant de réaliser la plupart des opérations courrantes d'un client HTTP, ainsi qu'une sérialization/desérialization JSON automatique
- Grace à la méthode *RestTemplate.getForEntity(x, x, x)* , faire un appel à l'API Github vers l'un de vos repositories: l'URL de base de l'API Github est https://api.github.com, auquel vous ajouterez le chemin « **/repos/{owner}/{repository_name} »** (voir https://docs.github.com/en/free-pro-team@latest/rest/reference/repos). Afin de recevoir une réponse dans un objet, vous devrez créer une classe Java (un DTO) permettant de mapper les

informations retournées par Github. Vous ajouterez l'annotation *@JsonIgnoreProperties(ignoreUnknown = true)* à cette classe

- Dans la classe GitRepositoryRepository, injectez le bean GitRepositoryDao.

III. L'assemblage



Voir https://github.com/Kaway/CSID-web-repository-base/blob/main/csid_schema.png

- Lors d'un appel à un GET */repository/{name}*, il faut:
  - ➔ appeler la base de données afin de récupérer les informations sur le repository
  - ➔ si la date de dernière mise à jour du repository est inférieure supérieur à 5mn, alors il faudra faire un appel à l'API REST de Github (grâce au GithubRepositoryDao) afin de récupérer les valeurs du nombre de issues et de forks, les enregistrer en base de données et retourner un objet Repository à jour au client; vous créerez les colonnes manquantes en base de données
- Ajouter un paramètre d'URL à la methode GET du controller (**@RequestParam**): ce paramètre va nous permettre de forcer l'update des informations de notre repository (un boolean suffira); si le boolean est à true, on appellera l'API REST Github même si les informations en base de données ont moins de 5m (on raffraichira toujours la base de données avec les informations de l'API REST Github). Le paramètre sera obligatoire (regarder la documentation de l'annation pour savoir comment faire)

IV. Bonus

- Rentre le temps de rafraîchissement configurable, en ajoutant une clé de configuration dans *application.properties* et en injectant sa valeur grâce à l'annotation ***@Value***