Travaux Dirigés – DUT 2e année

Initiation à Spring avec Spring-boot

Création d'une API REST

Introduction :

- Cloner le projet à l'adresse suivante: https://github.com/Kaway/dut-enterprise

  Le cours associé à notre module est disponible en v1 à l'adresse suivante : https://github.com/Kaway/dut-enterprise/blob/master/introduction_spring_1.pdf. Il y a beaucoup de choses dans le cours et il n'est pas encore complet, mais la partie sur REST et sur **spring-data-jpa** seront utiles pendant le module.

- Importer le projet dans votre IDE favori (je conseille Intellij Idea Community Edition[i] – mais Eclipse fera aussi l'affaire, je serai juste plus lent pour vous aider à le configurer)
  Le projet est basée sur version 11 de Java. Si Java 11 n'est pas disponible, il faut :
  - modifier le fichier pom.xml et remplacer la propriété «*java.version*» par 1.8
  - dans la classe «***Employee***», dans la méthode «***public Employee mergeWith(Employee other)***» il faut remplacer les appels à «***Objects.requireNonNullElse***» par une expresion ternaire renvoyant «***other.lastName***» s'il n'est pas ***null*** ou sinon «**lastName**»

- Télécharger Postman[ii] (https://www.getpostman.com/downloads/) et importer la collection https://github.com/Kaway/dut-enterprise/blob/master/DUT-2A_employee.postman_collection.json

- Tester le bon fonctionnement de la collection importée

Partie I : Employee

- Développer la méthode correspondant à la requête «*Create employee batch*»

- Rajouter les méthodes dans les controller, service et repository afin de récupérer la liste des employées triée par prénom puis nom («*John SMITH*» arrive devant «*Roger SMITH*»)

- Rajouter une adresse à notre employé : dans les classes «*Employee*» et «*EmployeeDto*», «*Address*» est un un objet. Dans la classe «*EmployeeEntity*», «*Address*» ne sera pas un objet, mais tous ses champs seront directement intégré à «*EmployeeEntity*»

Partie II : Salary

On souhaite verser un salaire à nos employés.

- Développer toute la chaîne Controller-Service-Repository et les objets associés afin de pouvoir enregistrer et lire des salaires (**GET** et **POST**). N'oubliez pas le batch pour verser des salaires à plusieurs employés en même temps (POST encore). On ne s'occupera pas du lien salaire-employé à cette question.
  Un salaire est composé du montant versé, du nombre de jours travaillée dans le mois, d'une date de versement ainsi que du mois/année de versement.
  Par défaut, on dira qu'un employé travaille 20 jours par mois.
  Attention, on en peut pas supprimer un salaire versé ni le modifier !
- Ajouter un employé à votre salaire. On utilisera **@ManyToOne** et **@OneToMany** dans «*SalaryEntity*» et «*EmployeeEntity*»
- Intégrer le fait qu'on ne puisse pas verser, pour le même mois, 2 fois le salaire d'un employé

Partie III : Vacation

On souhaite autoriser un employé à prendre des vacances !

- Développer toute la chaîne Controller-Service-Repository et les objets associés afin de pouvoir enregistrer, lire, supprimer et modifier les vacances d'un employé (GET, POST, PUT, PATCH, DELETE).
  On pourra ajouter des jours de congés 1 à 1 ou en mode batch (par exemple 2 semaines d'un coup). Dans le cas du mode batch, on pensera bien à ne pas compter les Samedis et Dimanches (on en gérera pas les jours fériés).
- Intégrer l'impossibilité d'ajouter un jour de congés en doublon pour un employé donné (attention à la gestion des intervalles)
- Modifier le versement du salaire : on doit maintenant prendre en compte les jours de congés posés durant le mois pour lequel on va verser le salaire

i Intelij Idea est un IDE, au même titre que Eclipse. Il possède une version open-source ainsi qu'une version payante. La version open-source est suffisante pour beaucoup d'utilisations

ii Postman est un logiciel (malheureusement pas open-source) permettant de requêter n'importe quel serveur HTTP. Il gère tous les verbes HTTP et vous permet de paramétrer finement tous les aspect de votre requête. Des alternatives (open-source) sont disponible, n'hésitez pas à les utiliser !