**TP API Rest**

**Vous aurez besoin à terme des ressources ici** [**https://github.com/mKapusta/rest-api.git**](https://github.com/mKapusta/rest-api.git)

**1/ Créer un projet Spring Boot**

Utiliser votre IDE pour le faire ou utilisez <https://start.spring.io/>

**2/ Créer les Objets représentant la base de données et une classe Database**

Créer des objets Professeur, Etudiant, Cours et Matiere.

Créer une classe Database contenant des listes static de chacun de ces éléments :

* 4 étudiants
* 2 professeurs
* 2 matières
* et quelques cours.

**3/ Créer un controller pour créer une ressource de Professeurs**

Vous devrez pour cela ajouter une dépendance spring-boot-starter-web.

Créer une classe ProfesseurController qui va proposer une ressource REST pour gérer les professeurs.

L’URL à laquelle le Controller devra répondre est “/professeurs”.

La classe contiendra une méthode pour chaque requête HTTP permettant de les manipuler :

* Lister tous les professeurs
* Récupérer un professeur par son identifiant
* Créer un professeur
* Modifier un professeur
* Supprimer un professeur

Chacune de ces méthodes aura une incidence sur la liste des professeurs dans la classe Database.

Vous aurez **AU MOINS** besoin de @RestController et @RequestMapping :-)

Vous pourrez tester le fonctionnement de votre Controller grâce à Postman, déjà utilisé lors de la manipulation des servlets

**Attention au verbe HTTP et à l’URL associés à chaque méthode!**

**4/ Recherche des Professeurs**

Il faut maintenant faire évoluer l’appel permettant de récupérer tous les professeurs. L’objectif est d’ajouter des paramètres à la requête entrante pour pouvoir filtrer par nom et prénom les professeurs.

@RequestParam vous permettra de récupérer un paramètre de l’URL. C’est l’équivalent de httpServletRequest.getParameter(...).

Vous en créerez donc 2, un pour le nom et un pour le prenom.

**N’oubliez pas de filtrer la liste avant de la retourner !**

**Vous avez créé votre premier moteur de recherche!**

Amélioration de la recherche : Spring permet de remplacer les @RequestParam par un objet. Créer donc un objet (ProfesseurSearchCriteria ?) pour contenir les deux propriétés permettant le filtre. Au sein de l’objet, pas besoin d’annotation. Spring fait tout automatiquement.

**5/ Gestion des exceptions**

Lorsqu’un élément n’existe pas (accès par identifiant, modification, etc), une bonne API Rest doit retourner une réponse avec un statut HTTP 404. **Ce n’est pas le cas de la recherche !**

A la différence de l’exercice sur Hibernate Validator, vous devrez ici créer vous même une exception soulevée manuellement par votre code.

Une exception est une classe Java qui étend la classe Exception.

Pour “transformer” l’exception en Réponse HTTP, n’oubliez pas l’annotation @ResponseStatus lors de la définition de l’Exception.

**6/ Utilisation d’une API Externe**

Nous allons ajouter le code postal de résidence de nos professeurs.

Pour ne pas avoir à gérer les fautes de frappes, seule le code postal sera stocké, et pas le nom de la ville.

Les services informatiques de l’état mettent à disposition publique une API permettant de récupérer des informations sur les communes françaises.

[**https://geo.api.gouv.fr/communes?codePostal=62440**](https://geo.api.gouv.fr/communes?codePostal=62440)

Retourne :

[

{

**"nom"**:"Harnes",

**"code"**:"62413",

**"codeDepartement"**:"62",

**"codeRegion"**:"32",

**"codesPostaux"**:[

"62440"

],

**"population"**:12524

}

]

Vous allez utiliser cette API pour récupérer les noms des villes et les retourner en même temps que les professeurs, **UNIQUEMENT DANS LE CAS DE LA REQUÊTE PAR IDENTIFIANT.**

Vous devrez donc commencer par :

7.1 : Créer un objet “Commune” qui contient les propriétés retournées par l’API publique.

7.2 : Utiliser un RestTemplate (new RestTemplate(...) + restTemplate.getForObject()) pour interroger l’API à partir du code postal. L’URL est disponible plus haut.

7.3 : Faire l’appel à l’API lors du traitement de la requête /professeur/{id} pour y ajouter la commune. Vous devrez pour cela ajouter une propriété à l’objet Professeur.

Vous **DEVEZ** lire l’url de l’API Publique depuis le fichier de properties. @Value est votre ami!

**7/ Ajout de contrôles sur les entrants**

En accord avec la programmation défensive, nous allons ajouter des contrôles sur les entrants de notre Controller :

* Le nom et le prénom ne doivent pas être Null ou vides.
* L’identifiant d’un professeur doit être supérieur ou égal à 1

Vous devrez pour cela ajouter Hibernate Validator(si nécessaire?) dans vos dépendances, valider le controller et le body de la création et de la mise à jour. (@Valid, @Validated,...)

**8/ Ajouter un controller pour gérer les Etudiants, les Cours, et les Matières**

Répétez le processus précédent pour les Etudiants et les Cours.

Inutile de faire une recherche pour les cours, seuls les Etudiants pourront être recherchés par noms et par prénoms, comme les professeurs.

Pour les matières, seule une méthode permettant de toutes les remonter suffira.

Inutile également de rappeler l’API des communes.