Cave à vin (partie 09)

# TP03 du module 04 – Spring Web Service

|  |
| --- |
| Ce TP sera un TP fil rouge du cours   * Réaliser un POST avec injection d’un objet JSON dans le body * Gérer les exceptions au travers d’un statut et d’un message |

|  |
| --- |
| **Durée estimée** |
| 30 minutes |

Contexte

En vous appuyant sur l’ensemble des concepts abordés dans le cours, mettez en place une application Spring Boot permettant la gestion d’une cave à vin.

L’application de cave à vin sera une application d’API Web sécurisée à la fin des itérations :

* Elle permet de gérer le stock des bouteilles de vins par le propriétaire qui est un administrateur de l’application
* Elle permet de gérer l’achat en ligne des bouteilles par un client
  + Les bouteilles de vin peuvent avoir 3 couleurs : Rouge, Blanc ou Rosé
  + Les bouteilles de vin sont associées aux régions françaises
* Un client et un propriétaire doivent s’authentifier
* Un visiteur est un client qui n’a pas de compte
* Il pourra uniquement consulter le stock

Dans les itérations nous réaliserons uniquement le back de l’application.

Cette application se fera avec Spring Boot.

Dans les itérations de ce module, nous allons développer l’API Web.

Solution

# Niveau 1 :

## Dans BouteilleController ; créer une méthode pour sauver une bouteille

* Capturer les exceptions de la BLL ; renvoyer un statut 406 et le message de l’exception

// Pour le Proprio

@PostMapping

**public** ResponseEntity<?> ajouterBouteille(@RequestBody Bouteille bouteille) {

**try** {

bService.ajouter(bouteille);

**return** ResponseEntity.*ok*(bouteille);

} **catch** (RuntimeException e) {

// Erreur BLL ou DAL

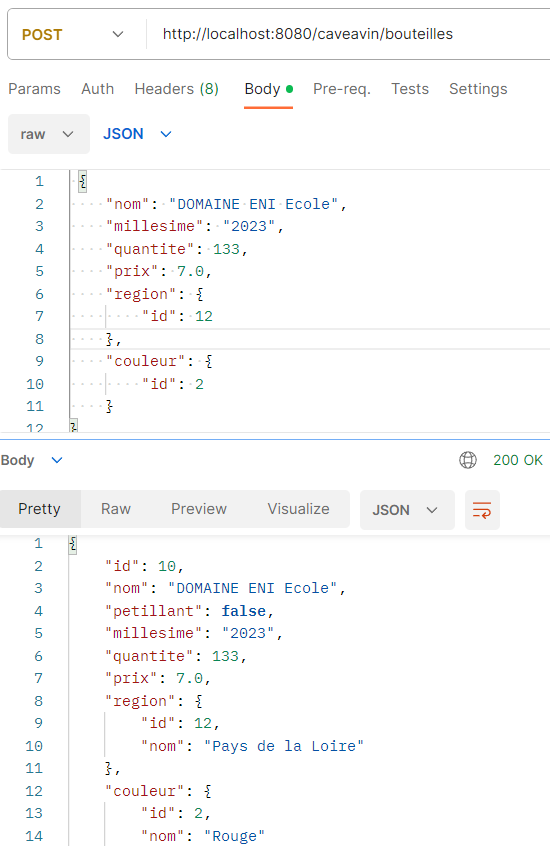
**return** ResponseEntity.*status*(HttpStatus.***NOT\_ACCEPTABLE***).body(e.getMessage());

}

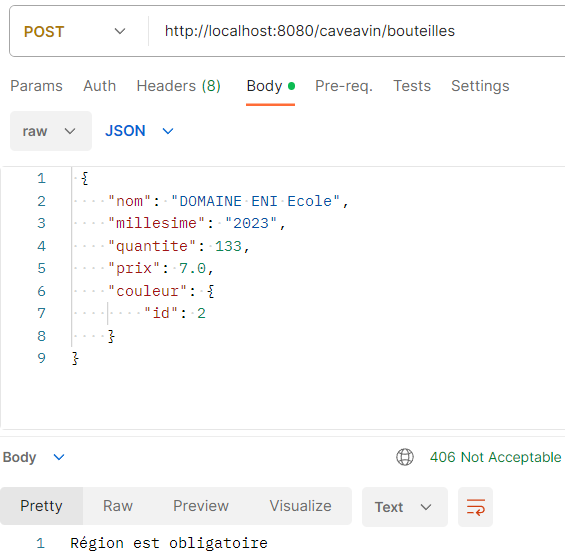
}

## Tester avec Postman

* Une bouteille avec sa couleur et sa région



* + Nous avons construit un objet JSON correct pour envoyer vers le serveur
    - Nous n’avons pas précisé le champ nom pour region et couleur
    - Ils ne sont pas indispensables
  + Le statut est 200
  + Et il y a eu ajout d’une nouvelle bouteille en base avec un identifiant généré
* Une bouteille avec sa couleur mais sans région



* + Il y a bien capture de RuntimeException provenant de la BLL
  + Et affichage du message : Région est obligatoire
* Evidemment, si vous tester avec des identifiants inconnus pour les couleurs ou les régions vous aurez aussi un message de la BLL

# Niveau 2 :

## Dans BouteilleController ; créer 2 méthodes

* Une pour mettre à jour une bouteille (PUT)
  + Elle ressemble beaucoup à la précédente,
  + Il faut en plus vérifier qu’il y a un identifiant non nul et >0
  + Si l’identifiant n’existe pas, cela créera une nouvelle bouteille comme précédemment
  + Capturer les exceptions de la BLL ; renvoyer un statut et le message de l’exception

// Pour le Proprio

@PutMapping

**public** ResponseEntity<?> miseAJourBouteille(@RequestBody Bouteille bouteille) {

**try** {

**if** (bouteille == **null** || bouteille.getId() == **null** || bouteille.getId() <= 0) {

**return** ResponseEntity.*status*(HttpStatus.***NOT\_ACCEPTABLE***)

.body("La bouteille et l'identifiant sont obligatoires");

}

bService.ajouter(bouteille);

**return** ResponseEntity.*ok*(bouteille);

} **catch** (RuntimeException e) {

// Erreur BLL ou DAL

**return** ResponseEntity.*status*(HttpStatus.***NOT\_ACCEPTABLE***).body(e.getMessage());

}

}

* L’autre pour supprimer une bouteille à partir de son identifiant (DELETE)
  + Capturer NumberFormatException, renvoyer un message et un statut 406
  + Capturer les exceptions de la BLL ; renvoyer un statut et le message de l’exception

// Pour le Proprio

@DeleteMapping("/{id}")

**public** ResponseEntity<?> deleteBouteille(@PathVariable("id") String idInPath) {

**try** {

**final** **int** idBouteille = Integer.*parseInt*(idInPath);

bService.supprimer(idBouteille);

**return** ResponseEntity.*ok*("Bouteille (" + idBouteille + ") est supprimée");

} **catch** (NumberFormatException e) {

// Statut 406 : No Acceptable

**return** ResponseEntity.*status*(HttpStatus.***NOT\_ACCEPTABLE***).body("Votre identifiant n'est pas un entier");

} **catch** (RuntimeException e) {

// Erreur BLL ou DAL

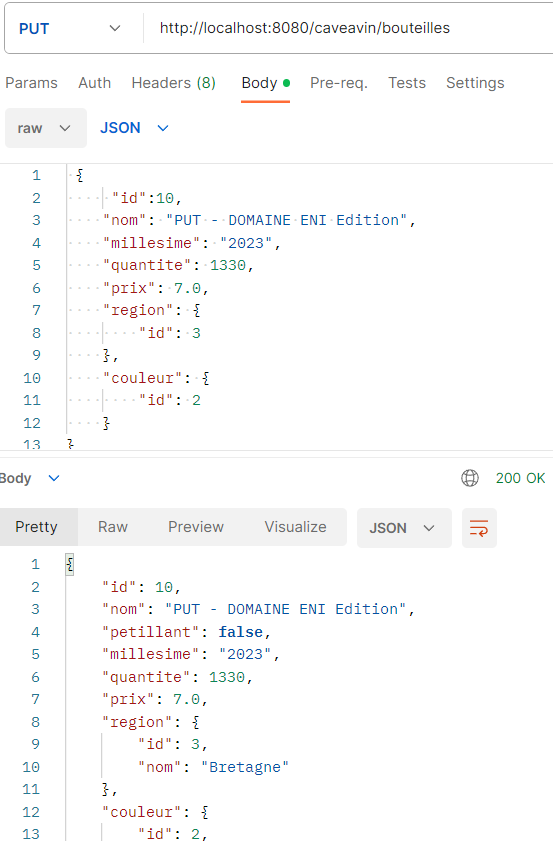
**return** ResponseEntity.*status*(HttpStatus.***NOT\_ACCEPTABLE***).body(e.getMessage());

}

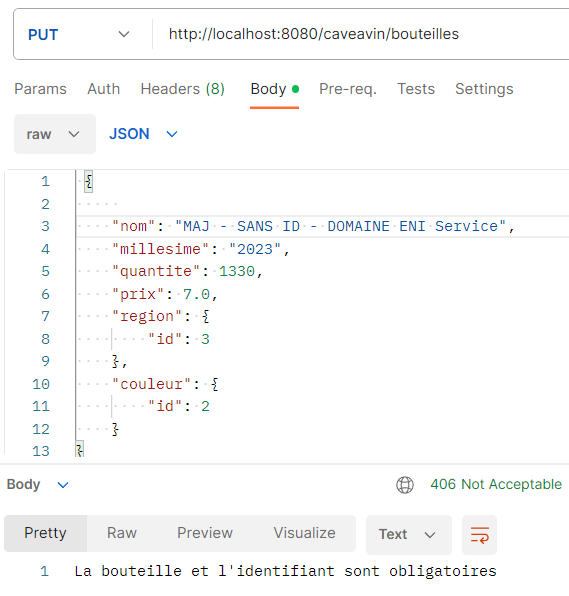
}

## Tester avec Postman

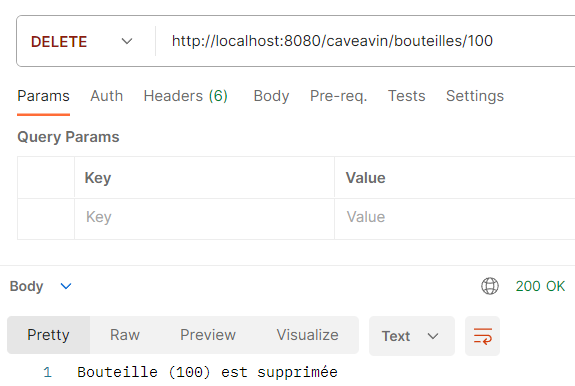
* Test du PUT
  + Avec des données correctes



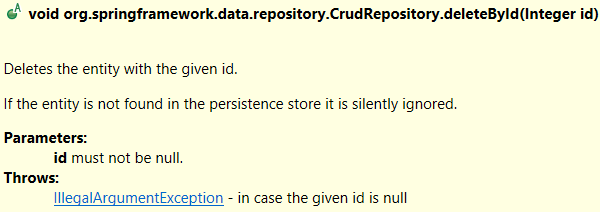
* + Avec aucun identifiant précisé



* + Si vous tester avec un identifiant qui n’existe pas. Spring Data JPA créera une nouvelle bouteille avec cet identifiant
* Test du DELETE
  + Avec un identifiant existant



* + Avec un identifiant inconnu
    - Remarquez que vous avez quand même la trace que la bouteille a été supprimée
    - Car Spring Data JPA ne lance pas d’erreur si aucune entité correspond



* + Avec un identifiant mal formé
    - Vous avez la levée de NumberFormatException traité par le contrôleur