Gestion des exceptions

# Démonstration 6 du module 4

|  |
| --- |
| Les objectifs de cette démonstration :   * Création d’un ControllerAdvice pour capturer les exceptions * L’utiliser sur des exceptions particulières et la classe mère Exception |

Contexte

* Nous continuons notre application représentant une partie du cœur de métier de l’ENI Ecole.

Déroulement

# Création AppExceptionHandler et gestion des Exception

* Création de la classe qui permet de capturer les exceptions des contrôleurs
* Positionner l’annotation @ControllerAdvice
* Dans un premier temps, création d’une méthode pour gérer toutes les Exception sans distinction :

**package** fr.eni.demo.exception;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.ControllerAdvice;

**import** org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice

**public** **class** AppExceptionHandler {

// Préciser les exceptions à gérer

@ExceptionHandler(value = { Exception.**class** })

**public** ResponseEntity<String> capturerException(Exception ex) {

**return** **new** ResponseEntity<String>(ex.getMessage(), HttpStatus.***NOT\_ACCEPTABLE***);

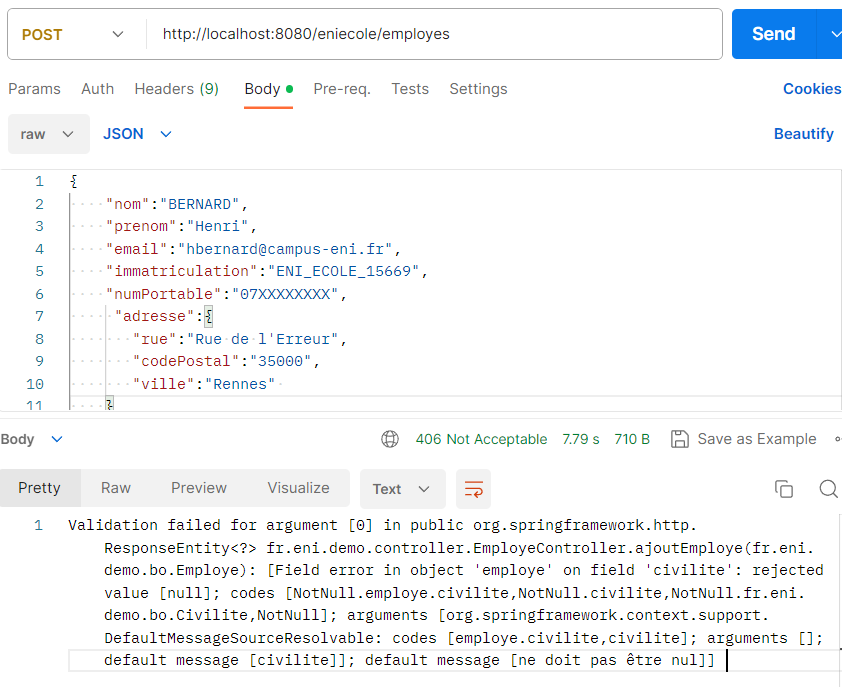
}

}

* Nous transmettons par défaut le message de l’exception Java
* Et un statut 406

## Exécution avec Postman

* Testons sans civilité :



* Le statut de la réponse (406)
* Nous avons bien maintenant, uniquement le contenu de l’attribut message d’Exception
  + Cependant, il y a encore trop d’information et pas seulement l’essentiel
  + En restant sur la classe mère pour toutes les exceptions le retour vers le client est trop généraliste
  + Dans ce cas, nous voudrions récupérer uniquement le message : « ne doit pas être nul »

# Gestion plus fine pour MethodArgumentNotValidException

* L’exception propagée par Spring pour une erreur de validation est MethodArgumentNotValidException
* Création d’une méthode pour gérer la capture de ces exceptions
* Et ajout d’un traitement plus fin pour retourner le message

…

**import** org.springframework.web.bind.MethodArgumentNotValidException;

…

// Gestion des exceptions de validation - MethodArgumentNotValidException

@ExceptionHandler(value = { MethodArgumentNotValidException.**class** })

**public** ResponseEntity<String> capturerMethodArgumentNotValidException(MethodArgumentNotValidException ex) {

String message = ex

.getFieldErrors()

.stream()

.map(e -> e.getDefaultMessage())

.reduce("Errors : ", String::concat);

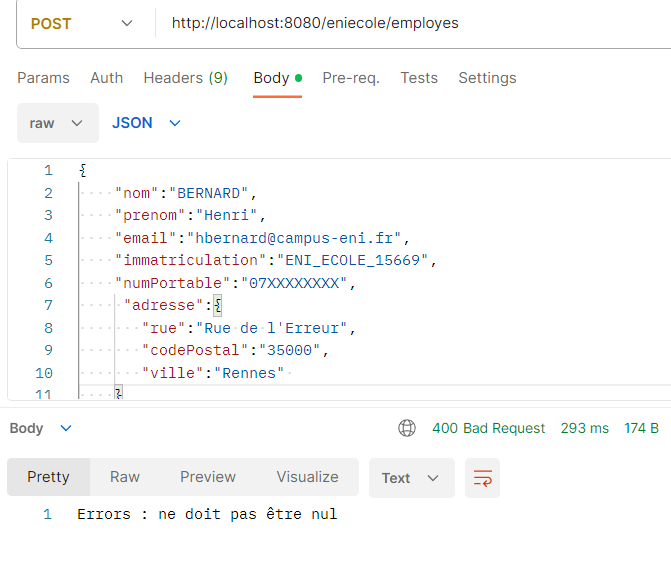
**return** ResponseEntity.*status*(HttpStatus.***BAD\_REQUEST***).body(message);

}

…

## Exécution avec Postman

* Testons sans civilité :



* Le statut de la réponse (400) 🡪 correspond bien au statut : HttpStatus.BAD\_REQUEST
* Nous avons bien maintenant, uniquement le contenu « getDefaultMessage »

Maintenant, nous avons de la capture d’exceptions au niveau BLL, Controller et Handler.

Cela évitera de transmettre des erreurs sans traitement vers le client.