Spring Data Rest

# Démonstration 01 du module 5

|  |
| --- |
| Les objectifs de cette démonstration sont   * Ajout du starter spring-boot-data-rest * Mise en place de Spring Data REST * Manipulation de Spring Data REST avec Postman |

Contexte

* Nous reprenons notre deuxième application, qui permettra de gérer les avis des stagiaires.
* Elle permettra plus tard, de réaliser des statistiques sur les cours.
* Notre application va se compléter au fil des démonstrations de ce module.
* Elle est indépendante de l’application précédente.
* Comme dans l’application précédente, les informations en base sont en anglais pour différencier les BO et les Document
* Il s’agit de l’application demo-no-sql
  + Si votre application n’est pas complète, vous pouvez reprendre le code Java dans les ressources
  + Attention, le paramétrage du projet Spring Boot n’est pas inclus
  + Il vous faut reprendre les démonstrations du module 3 pour valider les starters de Spring Boot
* Nous allons gérer les Avis, Formateur et Cours de notre application au travers de Spring Data REST
  + Cette partie de l’application sera directement disponible grâce à Spring Data REST.
  + Nous pouvons réaliser cette API car il n’y a pas de métier pour le moment d’associé

Repository

Spring Data REST



Spring Data

DB

Déroulement

# Configuration de Spring Data REST dans build.gradle

* Ajout du starter Spring Data REST

dependencies {

//Starter API REST

implementation 'org.springframework.boot:spring-boot-starter-data-rest'

* Faire « Refresh Gradle Project »

# BO :

Nous avons déjà une couche BO qui met en place les annotations de Spring Data. Il n’y a rien à faire à ce niveau pour le moment.

# DAL :

Nous avons déjà une couche DAL qui déclarer pour chaque Document un Repository de Spring Data. Il n’y a rien à faire à ce niveau non plus.

# Contrôleur de l’API REST :

* C’est le module Spring Data REST qui le fournit.
* Il permettra de manipuler des requêtes sur toutes les méthodes du protocole http
* Et même d’utiliser toutes les méthodes de MongoRepository (pagination, tri, …)
* Par défaut Spring utilise le nom de la classe au pluriel en anglais pour créer l’alias des urls.

# Exécution avec Postman

* Lancer l’application de votre serveur.
* Lancer Postman

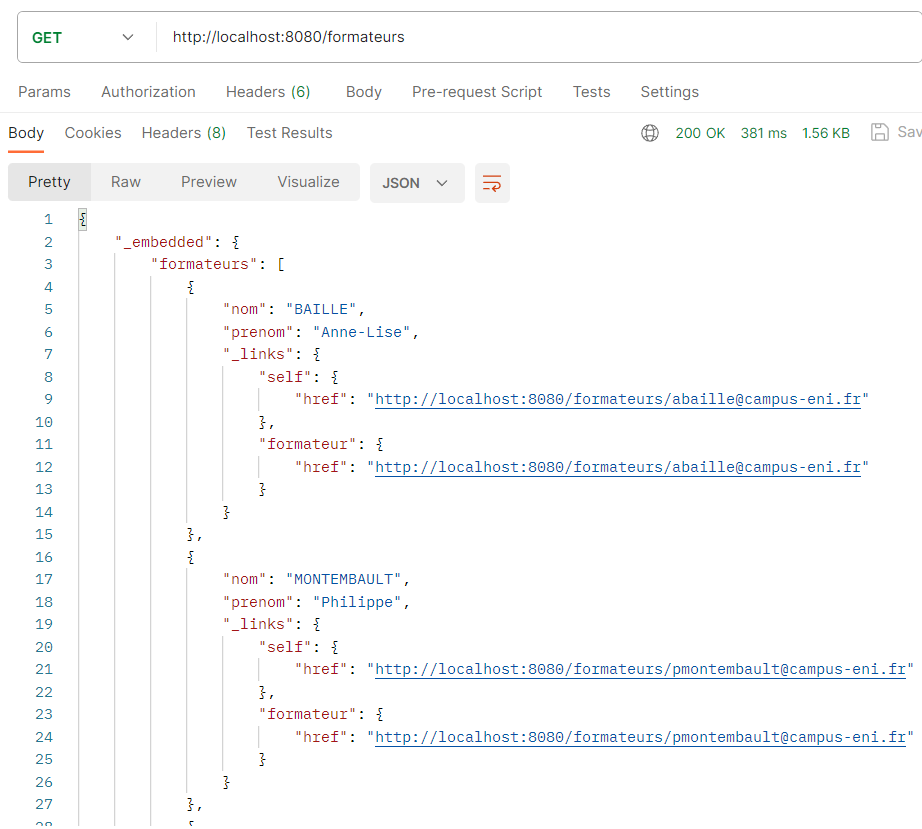
## FormateurRepository

* D’après Spring Data REST, l’URL pour manipuler la classe Formateur sera :

http://localhost:8080/formateurs

### GET http://localhost:8080/formateurs

* Dans Postman, cliquer pour créer une nouvelle requête GET dessus

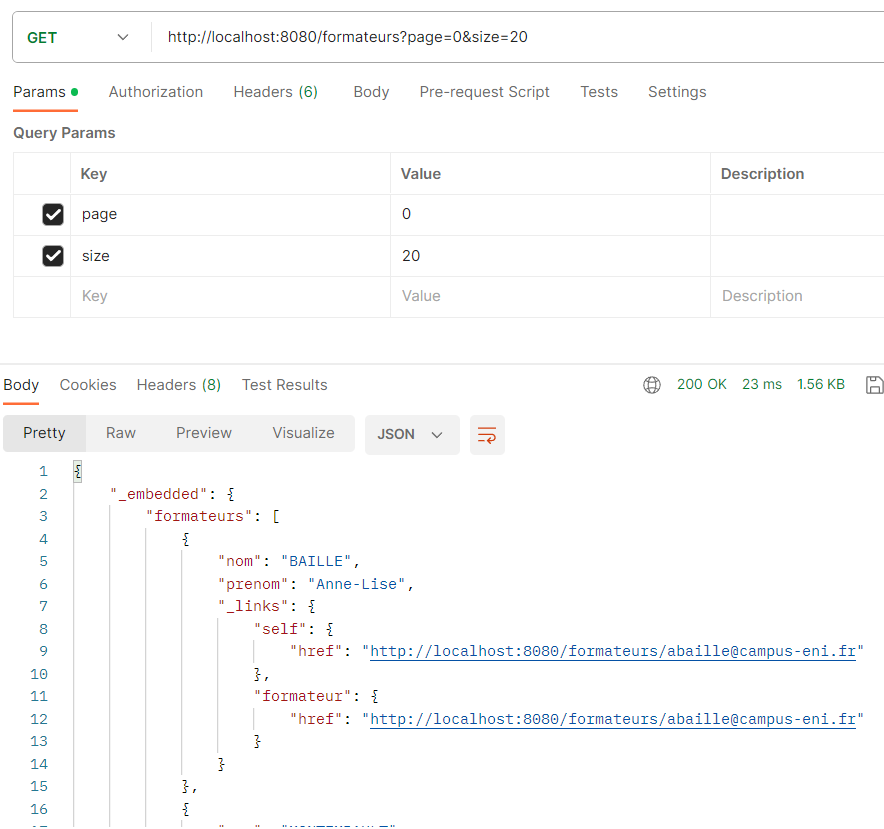


* Spring Data REST a généré le contrôleur pour répondre à la demande
* L’utilisation de l’API est assez simple et intuitive.
* De plus, les informations HATEOAS fournies par Spring Data REST vous permettent de cibler et d’automatiser l’utilisation de l’API par les clients.
  + Dans les informations présentées, nous avons directement l’URL pour manipuler un Formateur par son identifiant
  + Si vous descendez tout en bas du body de la réponse



* + Remarquez qu’il nous informe :
    - Du nombre d’éléments total : 3
    - Du nombre de pages : 1
    - Le nombre d’éléments par page : 20
  + Il propose 2 URLs :
    - Le premier permet de gérer la pagination et le nombre d’éléments par page
    - Le deuxième, permet de voir le profil du Formateur au format ALPS

### Pagination : http://localhost:8080/formateurs?page=0&size=20



* Par défaut cette requête nous remonte tous les Formateur qui sont contenus dans la page 0 avec 20 Max
  + Comme il n’y a que 3 Formateur ; ils sont tous là
* Essayons en changeant les paramètres :
  + Le nombre de formateur par page = 2
  + Et afficher la page 1
  + http://localhost:8080/formateurs?page=1&size=2

{

    "\_embedded": {

        "formateurs": [

            {

                "nom": "DELACHESNAIS",

                "prenom": "Frédéric",

                "\_links": {

                    "self": {

                        "href": "http://localhost:8080/formateurs/fdelachesnais@campus-eni.frr"

                    },

                    "formateur": {

                        "href": "http://localhost:8080/formateurs/fdelachesnais@campus-eni.frr"

                    }

                }

            }

        ]

    },

    "\_links": {

        "first": {

            "href": "http://localhost:8080/formateurs?page=0&size=2"

        },

        "prev": {

            "href": "http://localhost:8080/formateurs?page=0&size=2"

        },

        "self": {

            "href": "http://localhost:8080/formateurs?page=1&size=2"

        },

        "last": {

            "href": "http://localhost:8080/formateurs?page=1&size=2"

        },

        "profile": {

            "href": "http://localhost:8080/profile/formateurs"

        }

    },

    "page": {

        "size": 2,

        "totalElements": 3,

        "totalPages": 2,

        "number": 1

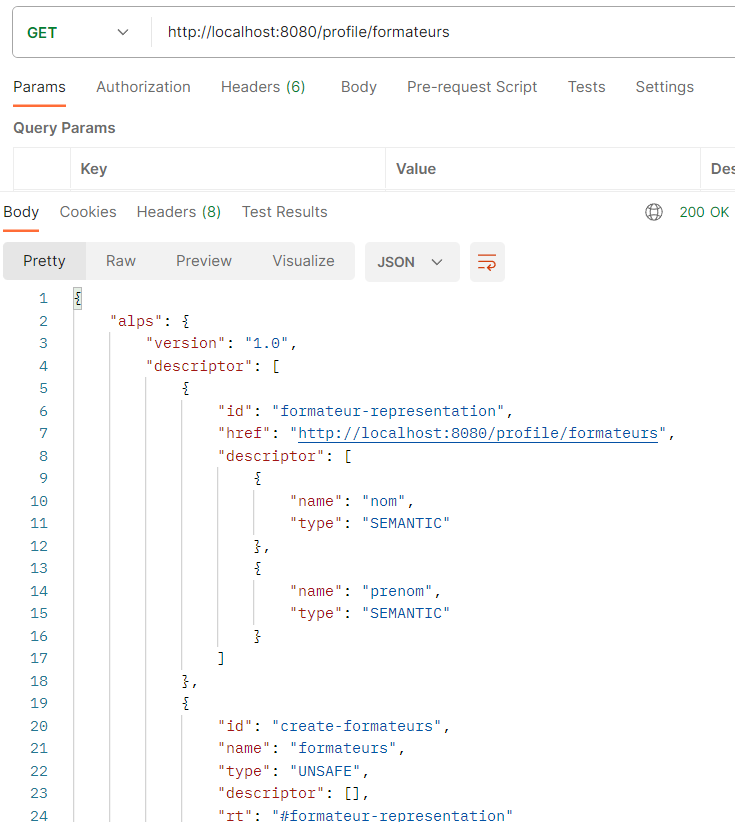
    }

}

* + Nous avons toujours au total : 3 Formateur
  + Mais 2 en nombre de pages
  + Et 2 Formateur par page
  + Actuellement, nous affichons la page 1, qui contient 1 seul Formateur
  + Remarquez qu’il nous propose les URL pour voir la première page, la page précédente, la courante, la dernière et le profil

Il est donc très facile de changer ces paramètres côté client.

### Profil : http://localhost:8080/profile/formateurs



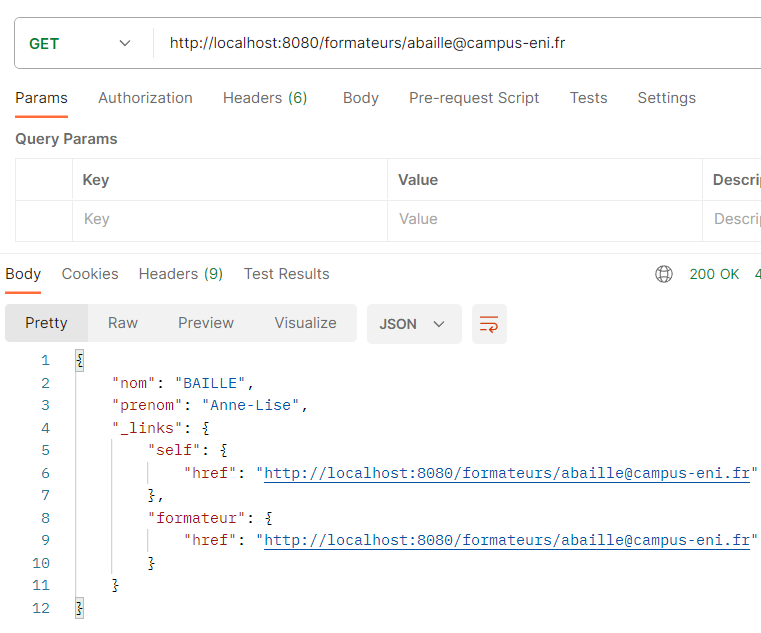
* ALPS est un format de données pour définir des descriptions simples de la sémantique au niveau de l'application.
  + Un document ALPS peut être utilisé comme profil pour expliquer la sémantique d'application d'un document avec un type de média indépendant de l'application (tel que HTML, HAL, Collection+JSON, Siren, etc.).
  + Cela augmente la possibilité de réutilisation des documents de profil sur tous les types de supports.
  + Cela aide les développeurs qui mettent en œuvre des solutions avec Spring

Ce n’est pas un élément de documentation, comme pourrait l’être Swagger.

Nous verrons cette documentation un peu plus loin

### Formateur par l’identifiant : http://localhost:8080/formateurs/abaille@campus-eni.fr

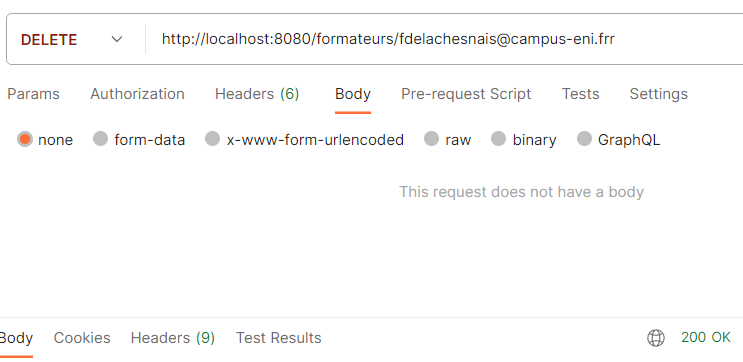
* Copier l’URL pour créer une nouvelle requête



* + Vous récupérez bien le détail d’un Formateur
* Il est possible de manipuler toutes les méthodes du protocole HTTP : GET, POST, PUT, PATCH et DELETE.
  + Il y a un formateur, qui a son email mal formé
* Regarder dans la liste des Formateur
  + L’URL pour DELACHESNAIS est http://localhost:8080/formateurs/fdelachesnais@campus-eni.frr
  + Car son email a « rr » à la fin
  + Email est la clef primaire, il faut donc supprimer ce formateur pour le recréer correctement

### DELETE Formateur par l’identifiant : http://localhost:8080/formateurs/fdelachesnais@campus-eni.frr

* Créer une nouvelle requête
  + Avec l’URL ci-dessus
  + En sélectionnant la méthode DELETE



* Vous pouvez vérifier soit en base soit en rappelant le Get général

### POST Formateur : http://localhost:8080/formateurs

* Créer une nouvelle requête
  + Avec l’URL ci-dessus
  + En sélectionnant la méthode POST
  + Et créer un objet Formateur dans le body avec les paramètres corrigé

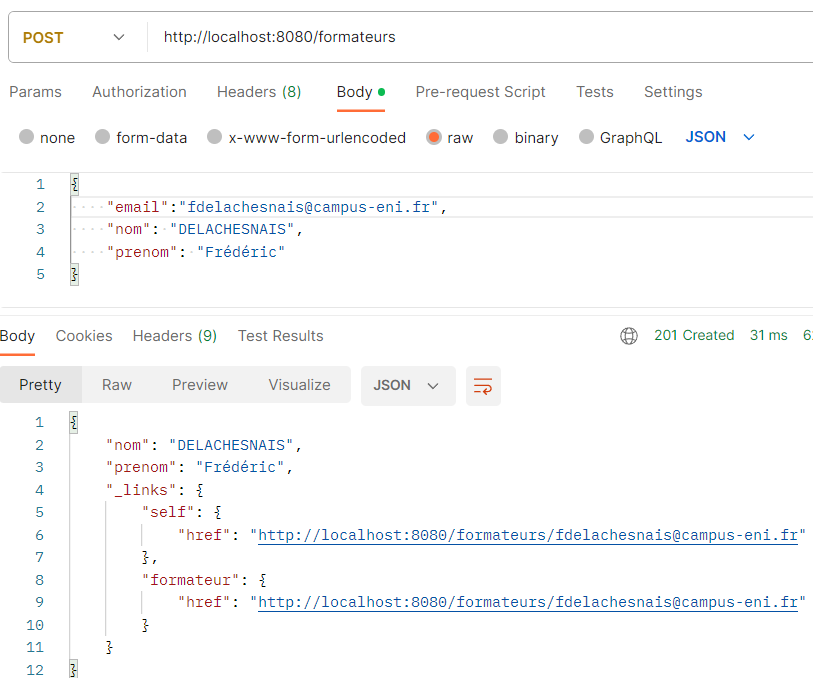
{

    "email":"fdelachesnais@campus-eni.fr",

    "nom": "DELACHESNAIS",

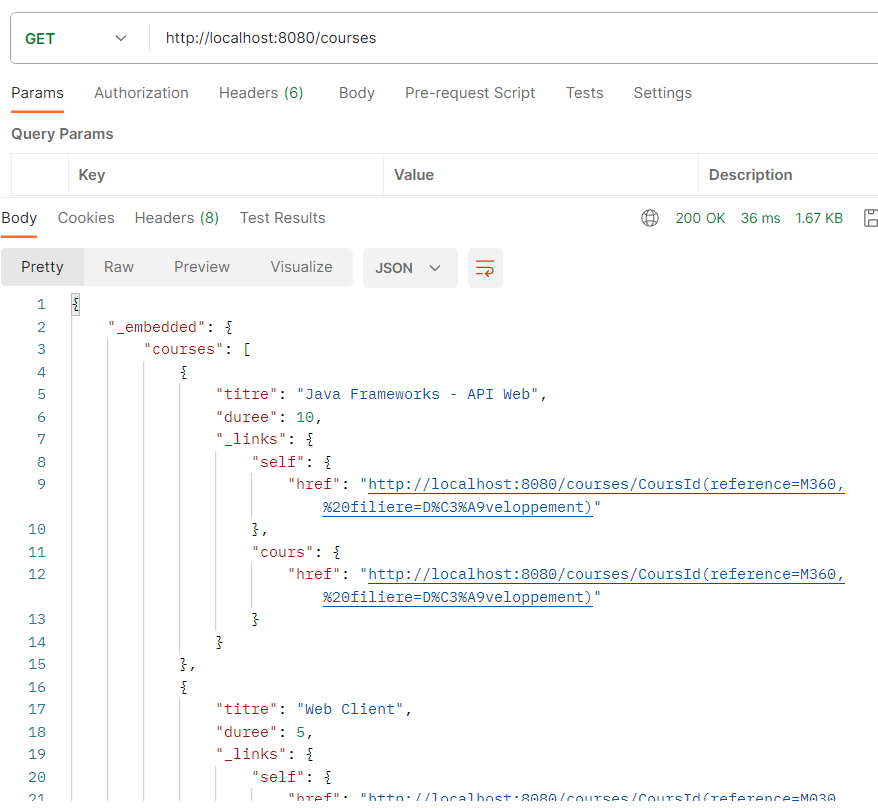
    "prenom": "Frédéric"

}



## CoursRepository

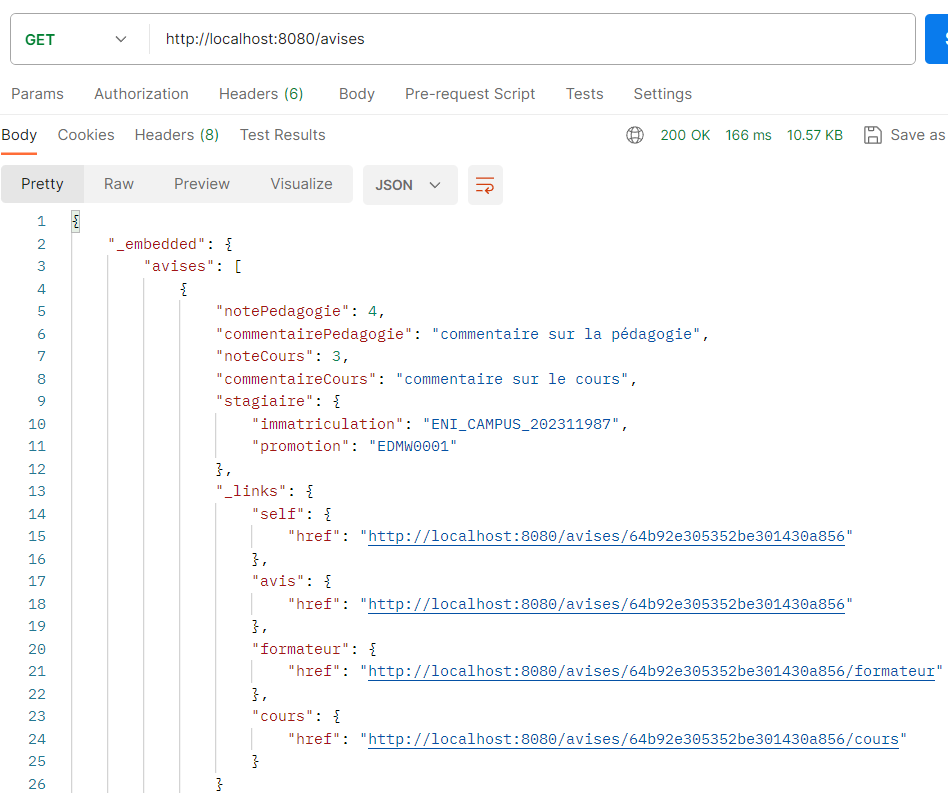
* Spring Data REST, va mettre au pluriel notre classe Cours
  + Pour lui, se sera donc courses
  + Pour nous cela devrait être cours, nous verrons comment préciser le bon URL par la suite
* Testons avec GET et http://localhost:8080/courses



* Spring Data REST, gère bien l’API depuis cet URL

## AvisRepository

* Spring Data REST, va mettre au pluriel notre classe Avis
  + Pour lui, se sera donc avises
  + Il y a donc le même problème
  + Si nous utilisions des mots en Anglais, il n’y aurait pas de souci
* Testons avec GET et http://localhost:8080/avises



* Spring Data REST, gère bien l’API depuis cet URL
  + Il nous affiche les URLs des associations vers Formateur et Cours

# Les tris

Tout comme la pagination, il est possible d’appeler les comportements de tris de Spring Data.

## GET http://localhost:8080/avises?sort=notePedagogie,desc

* Tri des Avis selon leur notePedagogie en descendant :

