Atelier 03 Spring Boot :

Développer des Web services REST avec Spring Boot

**Objectifs :**

1. Créer le Web service REST permettant de retourner tous les produits,
2. Créer le Web service REST permettant de consulter un produit,
3. Créer le Web service REST permettant de créer un produit,
4. Créer le Web service REST permettant de modifier un produit,
5. Créer le Web service REST permettant de supprimer un produit,
6. Créer le Web service REST permettant de retourner la liste des produits ayant une catégorie donnée,
7. Utiliser Spring Data REST *@RepositoryRestResource*,
8. Retourner l’ID avec *Spring Data REST*,
9. Restreindre les données avec les Projections.

## Créer le Web service REST permettant de retourner tous les produits

1. Créer, la classe ProduitRESTController dans le package com.example.demo.restcontrollers sont code est le suivant :

**package** com.example.demo.restcontrollers;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.web.bind.annotation.CrossOrigin; **import** org.springframework.web.bind.annotation.PathVariable; **import** org.springframework.web.bind.annotation.RequestBody; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RequestMethod; **import** org.springframework.web.bind.annotation.RestController; **import** com.nadhem.produits.entities.Produit;

**m**

@RestController @RequestMapping("/api") @CrossOrigin

**public class** ProduitRESTController { @Autowired

ProduitService produitService;

@RequestMapping(method = RequestMethod.***GET***)

**public** List<Produit> getAllProduits() {

**return** produitService.getAllProduits();

}

}

Au niveau de l’entité Categorie, ajouter l’annotation ***@JsonIgnore*** au-dessus de l’attribut produits, et ce pour éviter les références croisées (une boucle infinie lors de la sérialisation).

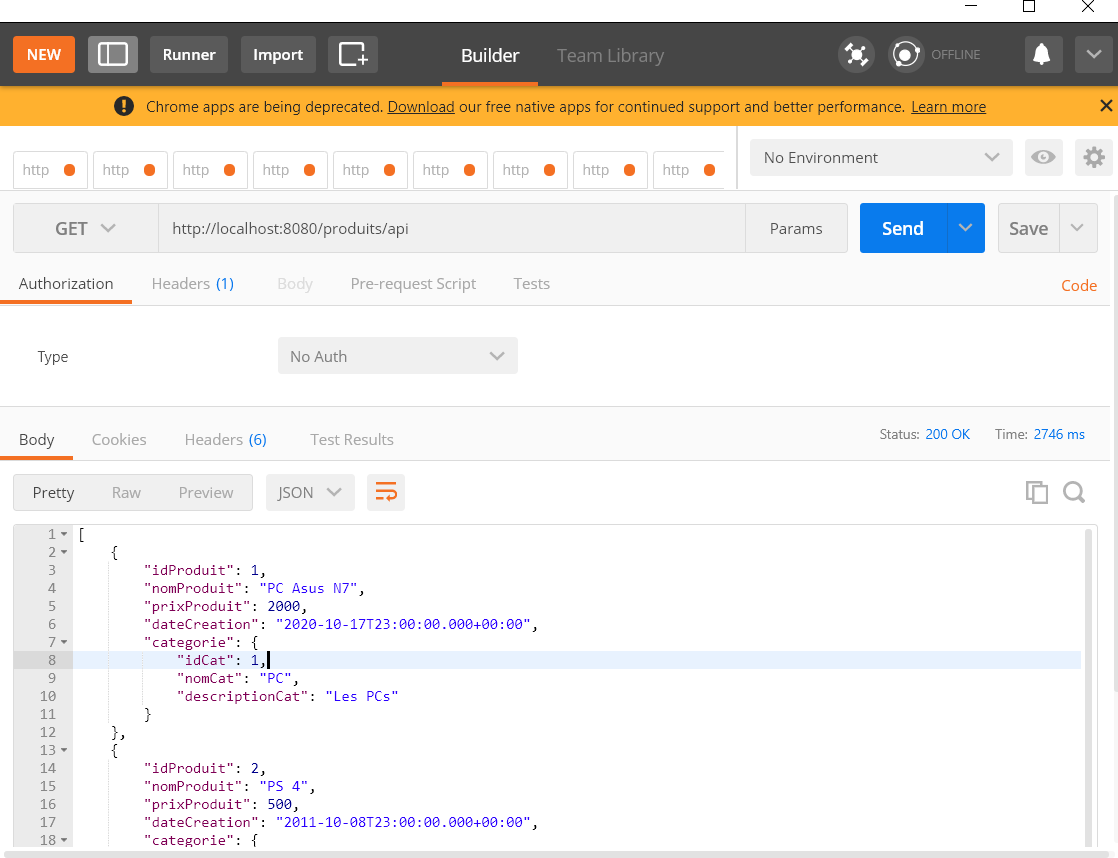
<http://localhost:8080/produits/api>

@JsonIgnore

@OneToMany(mappedBy = "categorie")

**private** List<Produit> produits;

1. Tester avec POSTMAN le web service REST :



## Créer le Web service REST permettant de consulter un produit

1. Ajouter, à la classe *ProduitRESTController*, la méthode *getProduitById* qui retourne un produit en acceptant son id :

@RequestMapping(value="/{id}",method = RequestMethod.***GET***)

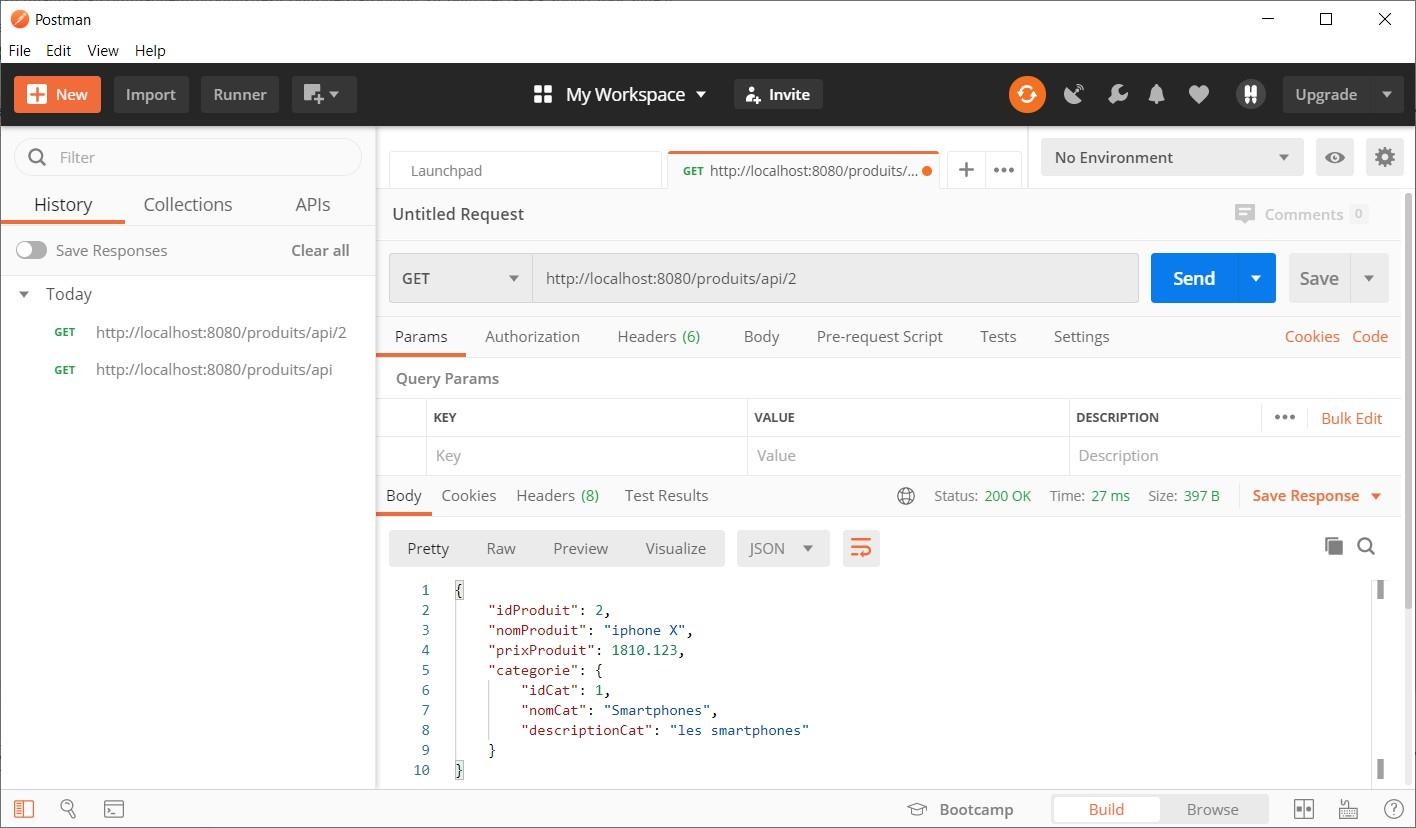
**public** Produit getProduitById(@PathVariable("id") Long id) {

**return** produitService.getProduit(id);

}

1. Tester avec POSTMAN le web service REST :

<http://localhost:8080/produits/api/2>



## Créer le Web service REST permettant de créer un produit

1. Ajouter, à la classe *ProduitRESTController*, la méthode *createProduit*

@RequestMapping(method = RequestMethod.***POST***)

**public** Produit createProduit(@RequestBody Produit produit) {

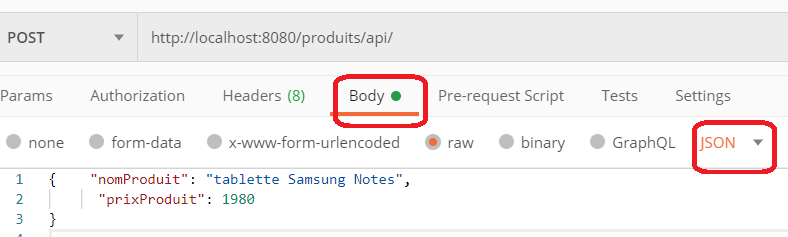
**return** produitService.saveProduit(produit);

}

1. Tester avec POSTMAN le web service REST :

<http://localhost:8080/produits/api>

* + Choisissez la méthode POST
  + Dans l’onglet Body, cliquez sur raw, puis entrer un produit au format JSON :



{ "nomProduit": "tablette Samsung Notes", "prixProduit": 1980

}

## Créer le Web service REST permettant de modifier un produit

1. Ajouter, à la classe *ProduitRESTController*, la méthode *updateProduit*

<http://localhost:8080/produits/api/>

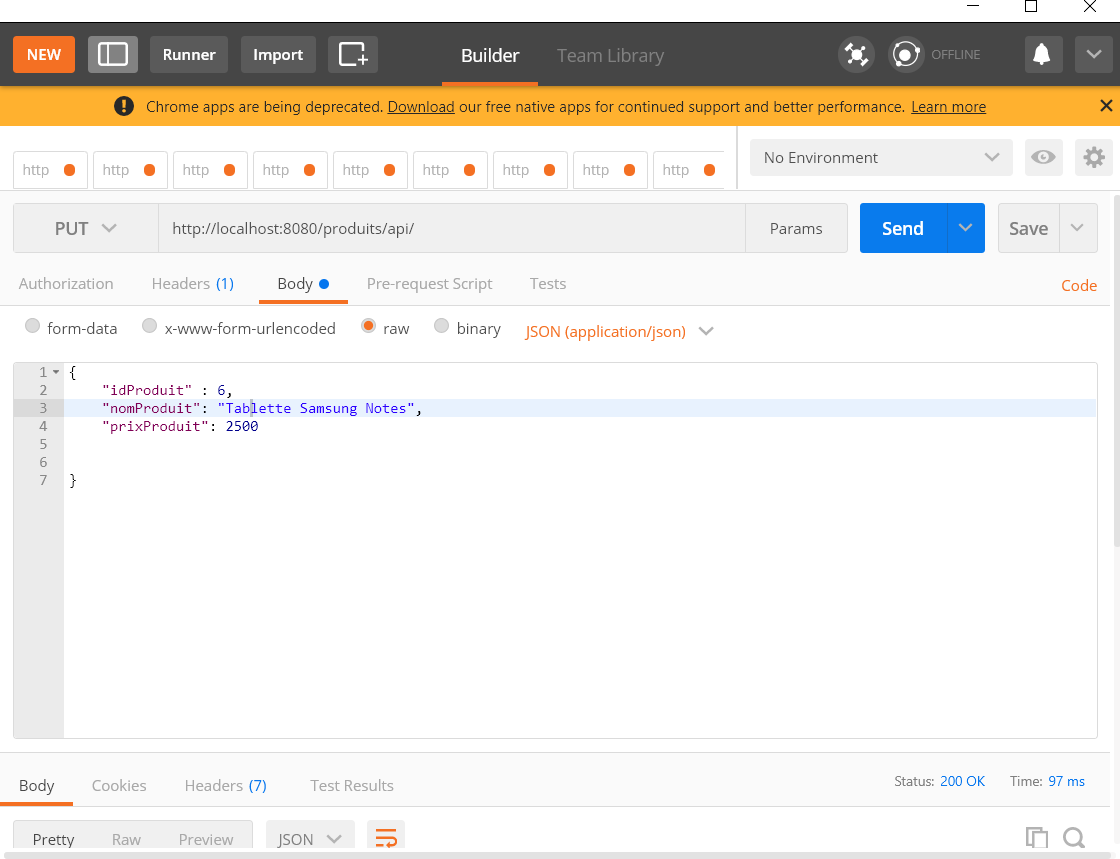
@RequestMapping(method = RequestMethod.***PUT***)

**public** Produit updateProduit(@RequestBody Produit produit) {

**return** produitService.updateProduit(produit);

}

1. Tester avec POSTMAN le web service REST :



## Créer le Web service REST permettant de supprimer un produit

1. Ajouter, à la classe *ProduitRESTController*, la méthode *deleteProduit*

<http://localhost:8080/produits/api/5>

@RequestMapping(value="/{id}",method = RequestMethod.***DELETE***)

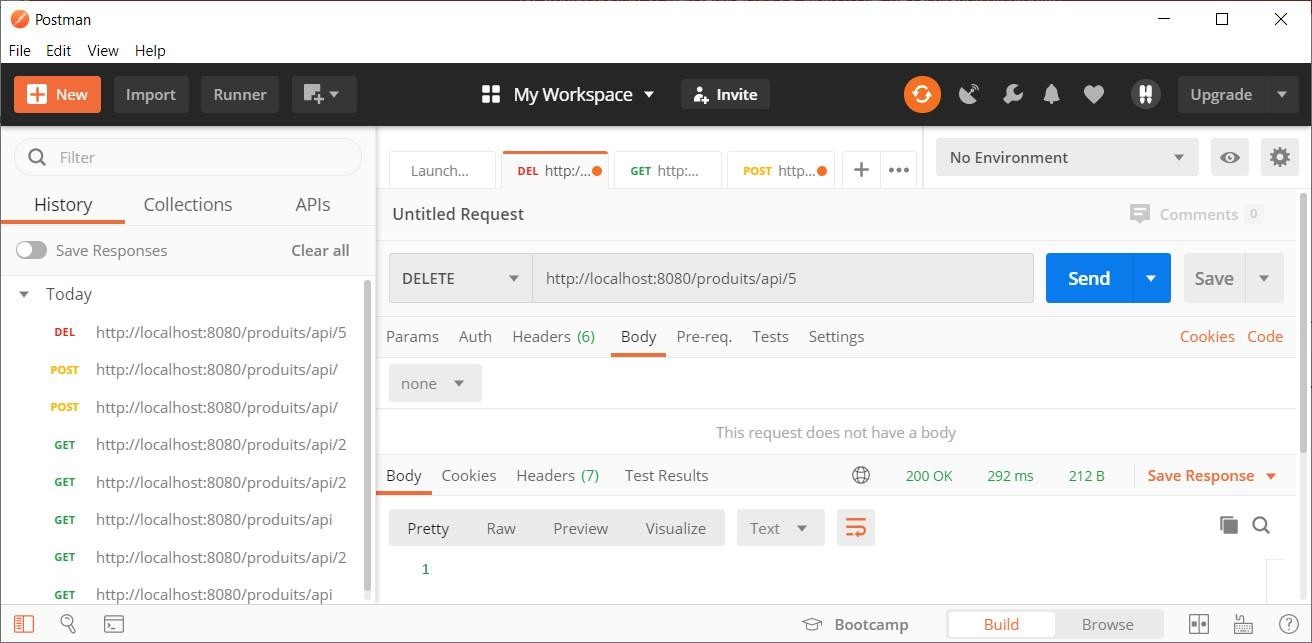
**public void** deleteProduit(@PathVariable("id") Long id)

{

produitService.deleteProduitById(id);

}

1. Tester avec POSTMAN le web service REST :



## Créer le Web service REST permettant de retourner les produits ayant une catégorie donnée

1. Ajouter, à la classe *ProduitRESTController*, la méthode *getProduitsByCatId*

@RequestMapping(value="/prodscat/{idCat}",method = RequestMethod.***GET***)

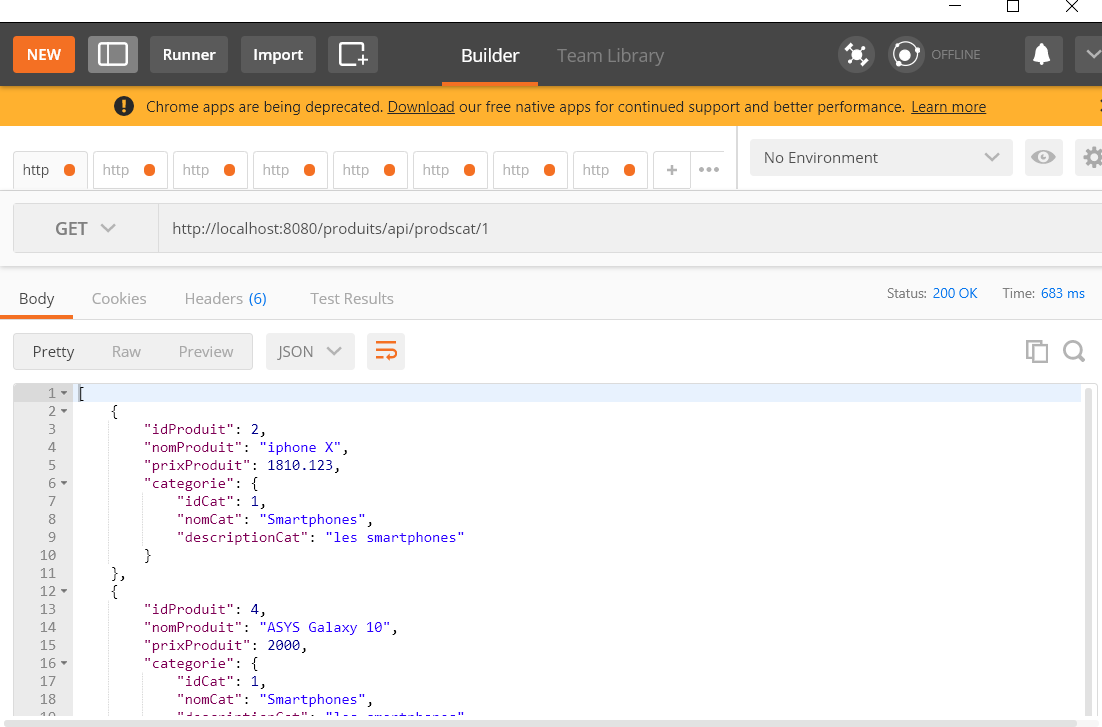
**public** List<Produit> getProduitsByCatId(@PathVariable("idCat") Long idCat) {

**return** produitService.findByCategorieIdCat(idCat);

}

1. Tester avec POSTMAN le web service REST :

<http://localhost:8080/produits/api/prodscat/1>



# Utiliser Spring Data REST @RepositoryRestResource

#### Avec Spring Data REST, on peut générer automatiquement tous les web services CRUD et autres

1. Ajouter la dépendance suivante au fichier pom.xml

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-rest</artifactId>

</dependency>

1. Ajouter l’annotation @RepositoryRestResource à l’interface ProduitRepository :

@RepositoryRestResource(path = "rest")

**public interface** ProduitRepository **extends** JpaRepository<Produit, Long> {…

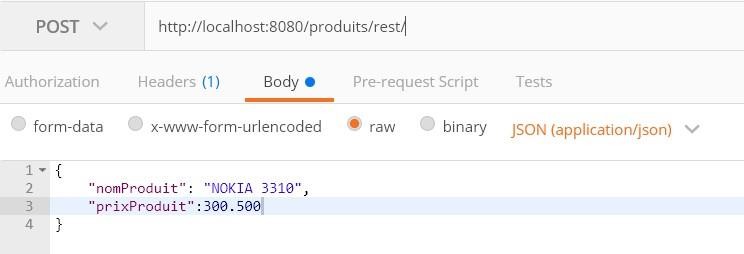
1. Tester avec POSTMAN les Web services suivants :

### Méthode GET :

* + <http://localhost:8080/produits/rest>
  + <http://localhost:8080/produits/rest/2>
  + <http://localhost:8080/produits/rest?size=2&page=0>
  + <http://localhost:8080/produits/rest?size=2&page=1>
  + [http://localhost:8080/produits/rest?sort=nomProduit,desc](http://localhost:8080/produits/rest?sort=nomProduit%2Cdesc)
  + [http://localhost:8080/produits/rest?size=2&page=0&sort=prixProduit,desc](http://localhost:8080/produits/rest?size=2&page=0&sort=prixProduit%2Cdesc)
  + <http://localhost:8080/produits/rest/search>
  + <http://localhost:8080/produits/rest/search/findByNomProduitContains?nom=PC>
  + <http://localhost:8080/produits/rest/search/findByCategorieIdCat?id=1>

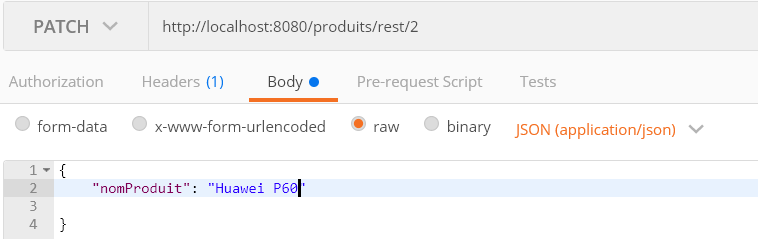
### Méthode POST :

* + - <http://localhost:8080/produits/rest>



### Méthode PATCH :

* + - <http://localhost:8080/produits/rest/2>



# Retourner l’ID avec Spring Data REST

Par défaut Spring Data REST ne retourne pas la propriéte ID. Or on peut avoir besoin de l’ID dans le résultat JSON si on utilise des frontend tels que Angular ou ReactJS. Pour retourner l’ID, on doit faire la configuration suivante :

1. Modifier la classe *ProduitsApplication* comme suit :

@SpringBootApplication

**public class** ProduitsApplication **implements CommandLineRunner** {

**@Autowired**

**private RepositoryRestConfiguration repositoryRestConfiguration;**

**public static void** main(String[] args) { SpringApplication.*run*(ProduitsApplication.**class**, args);

}

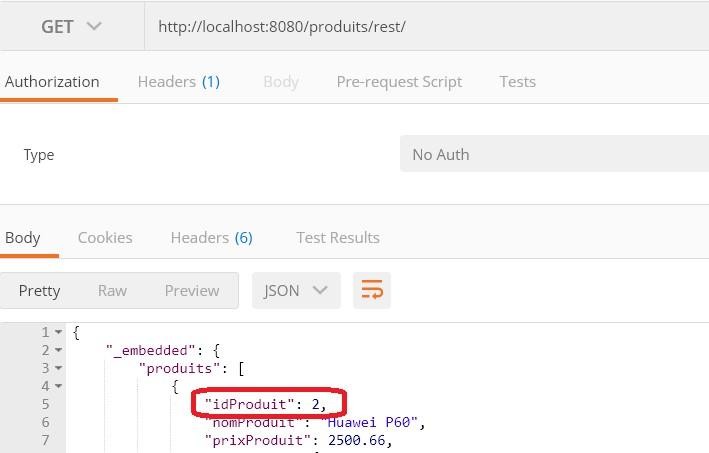
**@Override**

**public void run(String... args) throws Exception { repositoryRestConfiguration.exposeIdsFor(Produit.class);**

**}**

}

1. Vérifier l’apparition de l’ID en testant avec POSTMAN le Web service suivant :

<http://localhost:8080/produits/rest>

# Restreindre les données avec les Projections

L’objectif des projections est de limiter le résultat JSON retourné à un certain nombre d’attributs. Par exemple on peut avoir besoin seulement de l’attribut nomProduit :

1. Créer dans le package entities l’interface *ProduitProjection*

**package** com.nadhem.produits.entities;

**import** org.springframework.data.rest.core.config.Projection; @Projection(name = "nomProd", types = { Produit.**class** })

**public interface** ProduitProjection {

**public** String getNomProduit();

}

1. Tester la projection "*nomProd*" :

<http://localhost:8080/produits/rest?projection=nomProd>

