STITUT SUPERIEUR DES ETUDES TECHNOLOGIQUES DE SFAX

Département Technologies de Informatique

Atelier SOA -TP11
Développer et tester un service web REST avec Spring Boot

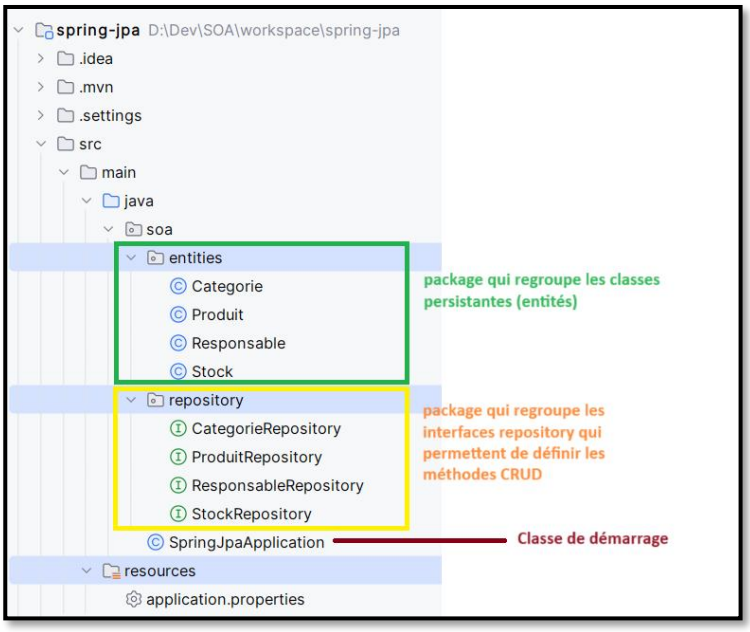
Objectifs

Développer et tester un Web Service REST Spring Boot

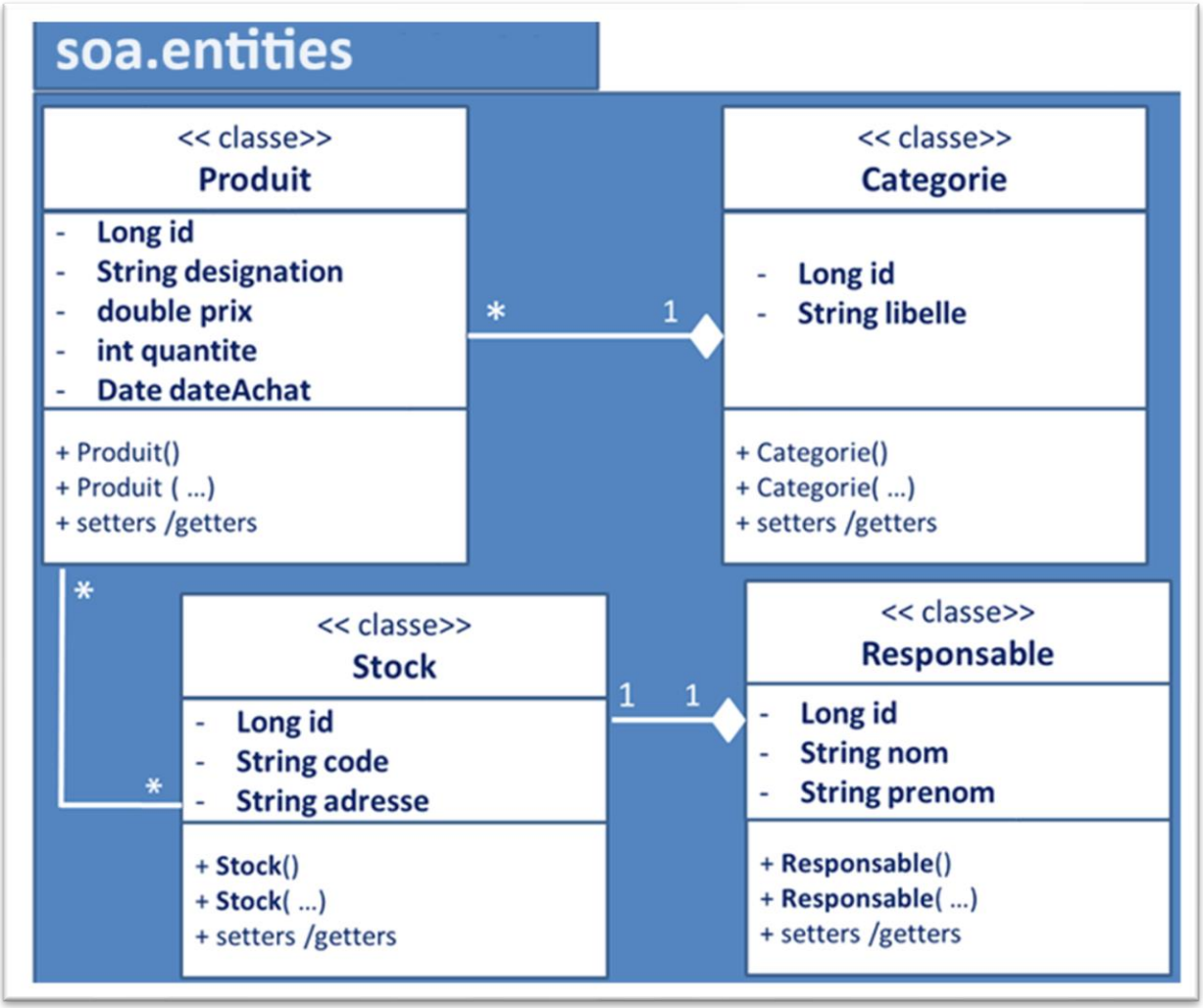
- **Créer un service Web REST**
  - Définir des paths avec des méthodes GET,POST,PUT et DELETE
- **Tester le service web avec « Advanced REST Client »**
  - Lancer une requête http avec ARC avec la méthode « GET »
  - Configurer ARC pour envoyer des requête avec les méthodes « POST », « PUT » et « DELETE »

A. Exécution d'un projet Spring Boot

1. Décompresser le fichier « spring-jpa.zip ».
2. Ouvrir avec **IntelliJ** le projet **spring-jpa**.
3. L'application présente deux packages : **soa.entities** et **soa.repository**



- Les classes persistantes sont conçues et modélisées selon le diagramme de classes persistantes :



- **Visualiser** les codes sources des classes persistantes et interpréter les annotations qui permettent d'indiquer les types des relations : **@OneToOne**, **@ManyToOne**, **@OneToMany**, **@ManyToMany**.

- Pour le package « **repository** », il s'agit d'un ensemble d'interfaces qui étends chacune l'interface de base « **JpaRepository** ». Chaque interface définit les méthodes CRUD de base et peut comporter des méthodes requêtes : (des méthodes qui se transforment en des requêtes SQL).

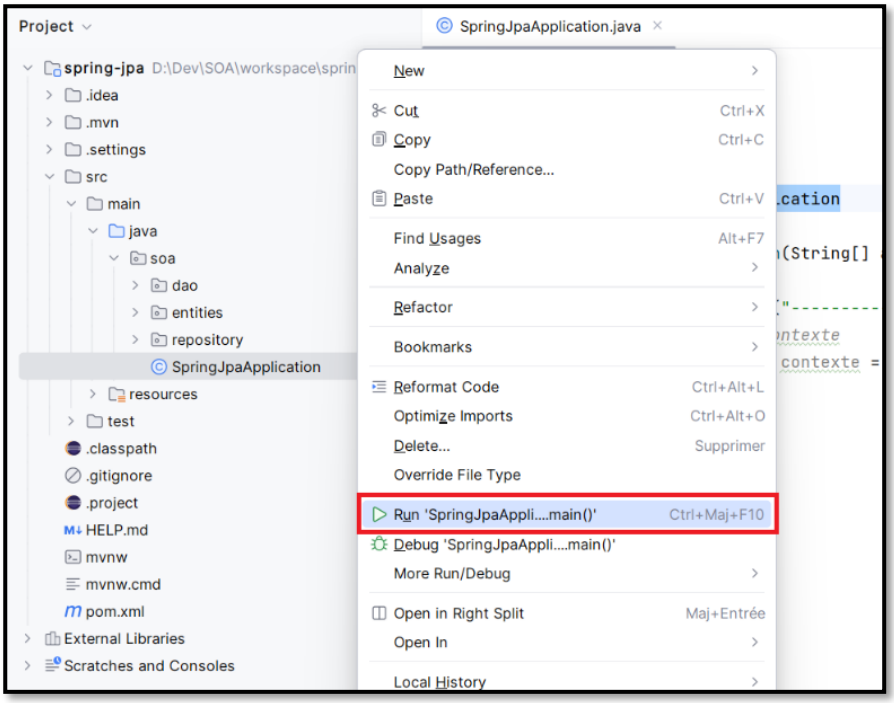
- Le fichier « application.properties » stocke les propriétés nécessaires pour la configuration :
  - Du serveur
  - De l'accès à la base de données
  - Du comportement du framework Hibernate
  - ....

application.properties

```
#database Configuration:
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/springSOABD?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=

#Hibernate Configuration:
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```

4. Lancer l'outil XAMPP et démarrer les deux modules **Apache** et **MySQL**.

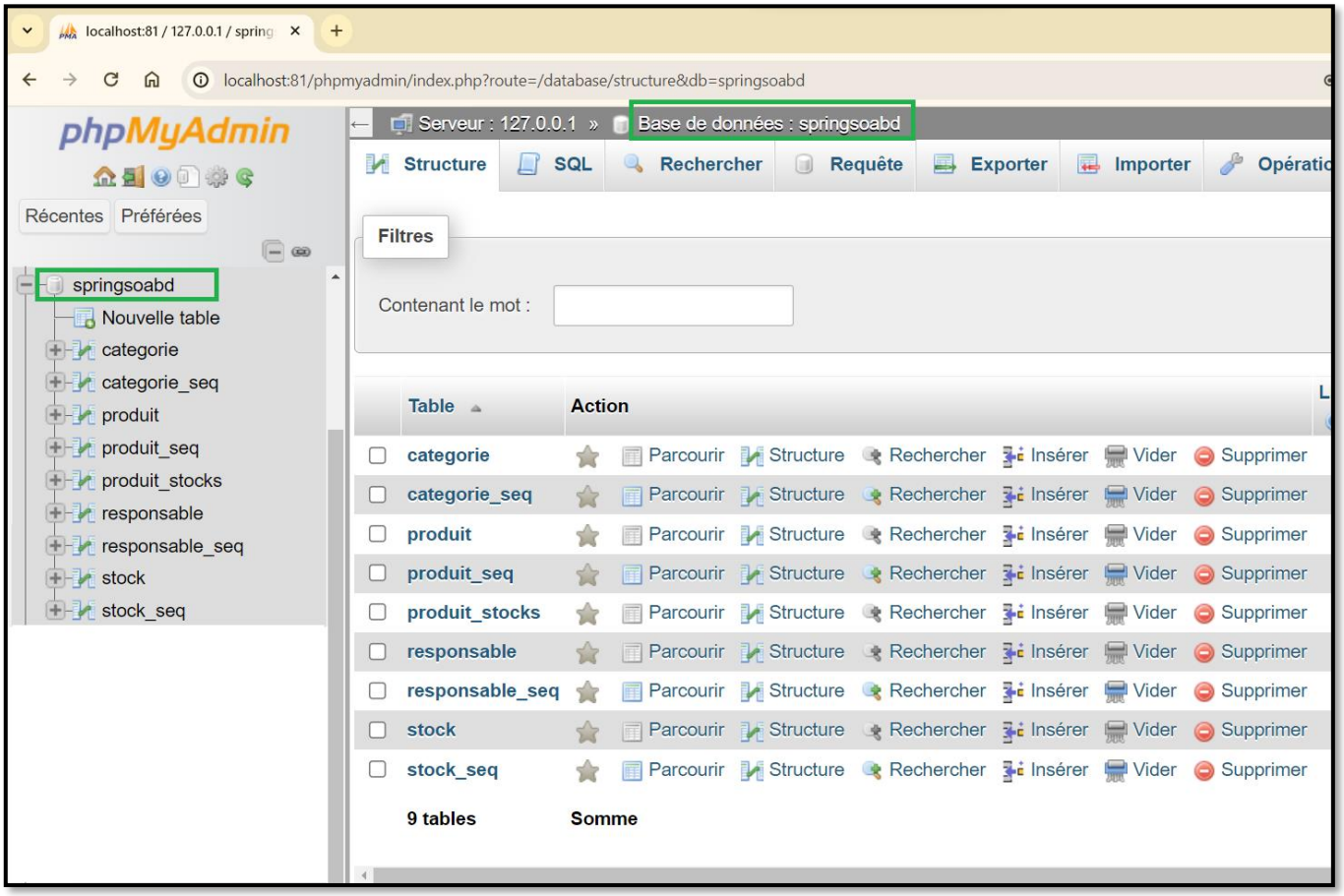
5. Lancer l'exécution de la classe **SpringJpaApplication** pour démarrer le moteur Spring et lancer le mécanisme d'ORM (q



6. Visualiser le résultat dans la console :

```
Initialized JPA EntityManagerFactory for persistence unit 'default'
Hibernate is in classpath; If applicable, HQL parser will be used.
Started SpringJpaApplication in 19.453 seconds (process running for 20.03)
Closing JPA EntityManagerFactory for persistence unit 'default'
HikariPool-1 - Shutdown initiated...
HikariPool-1 - Shutdown completed.
```

7. Au niveau relationnel, visualiser la création de la base de données « **springsoabd** » et les tables générées par le mécanisme ORM :



B. Réalisation de traitements sur les données avec des « repository »

8. Prendre la nouvelle version (donnée en pièce jointe) de la classe « **SpringJpaApplication** » qui utilise, directement, des objets « **repository** » pour passer les requêtes suivantes à la base de données :
    1) **Insertion de deux catégories**
    2) **Insertion de trois produits**
    3) **Changer la catégorie du Yaourt à Plastique**
    4) **Suppression de la catégorie Plastique**
    5) **Ajouter un nouveau produit 'Pain' à la catégorie 'Alimenatire'**
    6) **Ajouter un nouveau produit 'Stylo' à une nouvelle catégorie 'Bureautique**
    7) **Ajouter 4 responsables et 4 stocks**
    8) **Changer les responsables des deux stocks de Tunis et de Gabès**
    9) **Affecter le produit chocolat à tous les stocks**
    10) **Affecter le produit stylo au stock du Gabès**

9. Lancer l'exécution de la classe « **SpringJpaApplication** » et visualiser le résultat dans la console :

```
1-Insertion de deux catégories----------
2-Insertion de trois produits----------
********************Début**********************
Afficher tous les produits...
**********************************************
 Produit [id=1, designation=Yaourt,  prix=0.4, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=3, designation=Panier,  prix=1.2, quantite=30, enPromotion=false, dateAchat=2023-05-15, categorie=Categorie [id=2, code=PL, libelle=Plastique], stocks=[]]
********************Fin**********************
********************Début**********************
Afficher tous les produits de la catégorie [1]
**********************************************
 Produit [id=1, designation=Yaourt,  prix=0.4, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
********************Fin**********************
3-Changer la catégorie du Yaourt à Plastique----------
********************Début**********************
Afficher tous les produits de la catégorie [1]
**********************************************
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
********************Fin**********************
4-Suppression de la categorie Plastique----------
********************Début**********************
Afficher tous les produits...
**********************************************
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
********************Fin**********************
5-Ajouter un nouveau produit 'Pain' à la catégorie 'Alimenatire'----------
********************Début**********************
Afficher tous les produits...
**********************************************
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=4, designation=Pain,  prix=230.0, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
********************Fin**********************
6-Ajouter un nouveau produit 'Stylo' à une nouvelle catégorie 'Bureautique'----------
********************Début**********************
Afficher tous les produits...
**********************************************
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=4, designation=Pain,  prix=230.0, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=5, designation=Stylo,  prix=0.4, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=3, code=S, libelle=BUREATIQUE], stocks=[]]
********************Fin**********************
7-Ajouter 4 responsables et 4 stocks----------
********************Début**********************
Afficher tous les responsables avec leurs stocks
**********************************************
Responsable [id=1, nom=Ben Saleh, prenom=Ali, stock=Stock [id=1, code=1, adresse=Tunis]]
Responsable [id=2, nom=Ben Ahmed, prenom=Omar, stock=Stock [id=2, code=2, adresse=Sousse]]
Responsable [id=3, nom=Sallemi, prenom=Samira, stock=Stock [id=3, code=3, adresse=Sfax]]
Responsable [id=4, nom=Zouari, prenom=Zied, stock=null]
********************Fin**********************
8-Changer les responsables des deux stocks de Tunis et de Gabès----------
********************Début**********************
Afficher tous les responsables avec leurs stocks
**********************************************
Responsable [id=1, nom=Ben Saleh, prenom=Ali, stock=Stock [id=4, code=4, adresse=Gabes]]
Responsable [id=2, nom=Ben Ahmed, prenom=Omar, stock=Stock [id=2, code=2, adresse=Sousse]]
Responsable [id=3, nom=Sallemi, prenom=Samira, stock=Stock [id=3, code=3, adresse=Sfax]]
Responsable [id=4, nom=Zouari, prenom=Zied, stock=Stock [id=1, code=1, adresse=Tunis]]
********************Fin**********************
9-Affecter le produit chocolat à tous les stocks----------
10-Affecter le produit stylo au stock du Gabes----------
********************Début**********************
Afficher tous les produits...
**********************************************
 Produit [id=2, designation=Chocolat,  prix=2000.0, quantite=5, enPromotion=false, dateAchat=2023-02-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[Stock [id=1, code=1, adresse=Tunis], Stock [id=2, code=2, adresse=Sousse], Stock [id=3, code=3, adresse=Sfax], Stock [id=4, code=4, adresse=Gabes]]]
 Produit [id=4, designation=Pain,  prix=230.0, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
 Produit [id=5, designation=Stylo,  prix=0.4, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=3, code=S, libelle=BUREATIQUE], stocks=[Stock [id=4, code=4, adresse=Gabes]]]
 Produit [id=6, designation=Pain,  prix=230.0, quantite=20, enPromotion=false, dateAchat=2023-04-15, categorie=Categorie [id=1, code=AL, libelle=Alimentaire], stocks=[]]
********************Fin**********************
```

C. Réalisation de traitements sur les données en utilisant des services métiers

10. Créer un nouveau package « **soa.metier** »

11. Copier dans ce package les composants métiers (donnés en pièce jointe) :

- **CategorieMetierInterface**
- **CategorieMetierImpl**
- **ProduitMetierInterface**
- **ProduitMetierImpl**

12. Copier, dans le package « soa », la classe « **SpringJpaApplication2** » (donnée en pièce jointe) qui utilise des objets « **metier** » pour passer les requêtes suivantes à la base de données :

1) **Insérer un produit 'Ordinateur' sans catégorie et sans stock**
2) **Insérer un produit 'Imprimante' avec la catégorie 'Informatique' et sans stock**
3) **Insérer un produit 'Tablette' avec la catégorie 'Jouet' et sans stock**
4) **Insérer un produit 'SmartPhone' avec la catégorie 'Informatique' et le Stock du 'Tunis'**
5) **Changer la catégorie du produit 'Tablette' à 'Informatique'**
6) **Rendre tous les produits achetés avant '2023' en promotion**
7) **Calculer le coût de vente de tous les produits en stock en appliquant la remise sur les produits en promotion**

13. Lancer l'exécution de la classe « **SpringJpaApplication2** » et visualiser le résultat dans la console :

```
-1-Insérer un produit 'Ordinateur' sans catégorie et sans stock ----------
**********Début**********
Afficher tous les produits...
**********
 Produit [id=1, designation=Ordinateur,  prix=2400.0, quantite=1, enPromotion=false, dateAchat=2021-04-15, categorie=null, stocks=[]]
**********Fin**********
-2-Insérer un produit 'Imprimante' avec la catégorie 'Informatique' et sans stock ----------
**********Début**********
Afficher tous les produits...
**********
 Produit [id=1, designation=Ordinateur,  prix=2400.0, quantite=1, enPromotion=false, dateAchat=2021-04-15, categorie=null, stocks=[]]
 Produit [id=2, designation=Imprimante,  prix=500.0, quantite=10, enPromotion=false, dateAchat=2022-02-20, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
**********Fin**********
-3-Insérer un produit 'Tablette' avec la catégorie 'Jouet' et sans stock ----------
**********Début**********
Afficher tous les produits...
**********
 Produit [id=1, designation=Ordinateur,  prix=2400.0, quantite=1, enPromotion=false, dateAchat=2021-04-15, categorie=null, stocks=[]]
 Produit [id=2, designation=Imprimante,  prix=500.0, quantite=10, enPromotion=false, dateAchat=2022-02-20, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
 Produit [id=3, designation=Tablette,  prix=300.0, quantite=3, enPromotion=false, dateAchat=2023-01-10, categorie=Categorie [id=2, code=JOUET, libelle=Jouets], stocks=[]]
**********Fin**********
-4-Insérer un produit 'SmartPhone' avec la catégorie 'Informatique' et le Stock du Tunis
**********Début**********
Afficher tous les produits...
**********
 Produit [id=1, designation=Ordinateur,  prix=2400.0, quantite=1, enPromotion=false, dateAchat=2021-04-15, categorie=null, stocks=[]]
 Produit [id=2, designation=Imprimante,  prix=500.0, quantite=10, enPromotion=false, dateAchat=2022-02-20, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
 Produit [id=3, designation=Tablette,  prix=300.0, quantite=3, enPromotion=false, dateAchat=2023-01-10, categorie=Categorie [id=2, code=JOUET, libelle=Jouets], stocks=[]]
 Produit [id=4, designation=SmartPhone,  prix=1000.0, quantite=2, enPromotion=false, dateAchat=2023-05-15, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[Stock [id=1, code=1, adresse=Tunis]]]
**********Fin**********
-5-Changer la catégorie du produit 'Tablette' à 'Informatique'
----Succès du changement de catégorie----'
**********Début**********
Afficher tous les produits...
**********
 Produit [id=1, designation=Ordinateur,  prix=2400.0, quantite=1, enPromotion=false, dateAchat=2021-04-15, categorie=null, stocks=[]]
 Produit [id=2, designation=Imprimante,  prix=500.0, quantite=10, enPromotion=false, dateAchat=2022-02-20, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
 Produit [id=3, designation=Tablette,  prix=300.0, quantite=3, enPromotion=false, dateAchat=2023-01-10, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
 Produit [id=4, designation=SmartPhone,  prix=1000.0, quantite=2, enPromotion=false, dateAchat=2023-05-15, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[Stock [id=1, code=1, adresse=Tunis]]]
**********Fin**********
-6-Rendre tous les produits achetés avant '2023' en promotion
**********Début**********
Afficher tous les produits...
**********
 Produit [id=1, designation=Ordinateur,  prix=2400.0, quantite=1, enPromotion=true, dateAchat=2021-04-15, categorie=null, stocks=[]]
 Produit [id=2, designation=Imprimante,  prix=500.0, quantite=10, enPromotion=true, dateAchat=2022-02-20, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
 Produit [id=3, designation=Tablette,  prix=300.0, quantite=3, enPromotion=false, dateAchat=2023-01-10, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[]]
 Produit [id=4, designation=SmartPhone,  prix=1000.0, quantite=2, enPromotion=false, dateAchat=2023-05-15, categorie=Categorie [id=1, code=INF, libelle=Informatique], stocks=[Stock [id=1, code=1, adresse=Tunis]]]
**********Fin**********
-7-Calculer le coût de vente de tous les produits en stock en appliquant la remise sur les produits en promotion
Cout de vente du stock:6600.0
```

D. Développement d'un service web REST

14. Prendre une copie du projet «**spring-jpa**» et le nommer «**spring-jpa-REST**» **et l'ouvrir avec IntelliJ.**

15. Mettre à jour la valeur de l'artifactId à « **spring-jpa-REST** » dans le fichier « **POM.xml** »

16. Référencer dans le fichier «**application.properties**» vers une nouvelle base de données «**springSoaRestBD**» .

17. Déclarer, dans le fichier «pom.xml», la dépendance du module **Spring Web** pour le développement des composants WEB nécessaires :

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

18. Créer, dans un nouveau package «**soa.controller**», la classe qui définit le service web REST et qui est nommée «**ProduitRESTController**» :

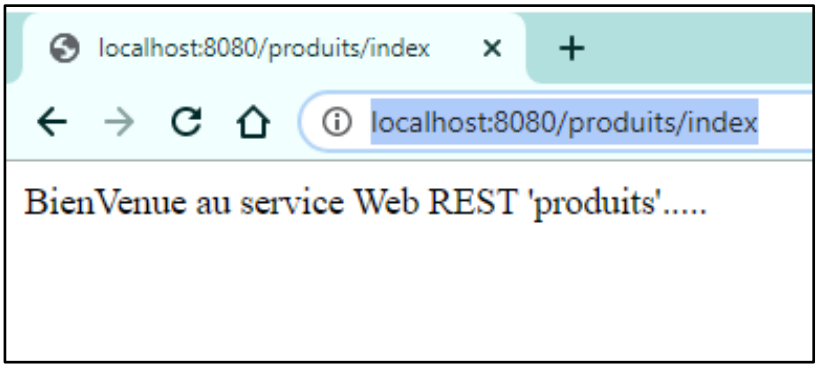
```
package soa.controller;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController // pour déclarer un service web de type REST
@RequestMapping("/produits")  //  relatif à  http://localhost:8080/produits
public class ProduitRESTController
{


       //  Message d'accueil
       //  http://localhost:8080/produits/index  (GET)
      @RequestMapping(value ="/index" ,method = RequestMethod.GET)
      public String accueil() {
            return "BienVenue au service Web REST 'produits'.....";
      }
}
```

19. Exécuter le projet (exécuter la classe « **SpringJpaApplication2** ») pour déployer le service web. Remarquer le démarrage du serveur Tomcat avec le port 8080 :

```
main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries ma
main] o.s.d.j.r.query.QueryEnhancerFactory     : Hibernate is in classpath; If applicable, HQL parser will be used.
main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
main] i.m.spring.JpaSpringBootApplication      : Started JpaSpringBootApplication in 4.084 seconds (process running for 4.779)
```

20. Aller au navigateur web pour afficher le message d'accueil en utilisant l'url suivant :

http://localhost:8080/produits/index



21.L'annotation « **@RequestMapping** » permet de spécifier :

- Le path à travers l'attribut « **value** »
- La méthode **GET** pour la requête http à travers l'attribut « **method** »

- Il est possible d'avoir le même comportement en remplaçant l'annotation «**@RequestMapping**» par l'annotation «**@GetMapping**» comme suit :

```
@GetMapping(value ="/index" )
```

- Enregistrer la modification et vérifier le résultat sur le navigateur web pour obtenir le même résultat dans la navigateur web.

E. Afficher des données avec la méthode GET au format JSON

22. Ajouter dans la classe «**ProduitRESTController**» une méthode « **getAllProduits** » qui permet de retourner la liste des produits :

- Elle porte le path « **/** »
- Utilise la méthode **GET** du http
- Utilise la méthode «**findAll**» de l'interface «**ProduitRepository**» pour récupérer tous les produits
- Et retourne les données sous le format **JSON** (par défaut)

Prendre cette nouvelle version du service web :

```
package soa.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import soa.entities.Produit;
import soa.repository.ProduitRepository;
```

```
@RestController // pour déclarer un service web de type REST
@RequestMapping("/produits")  //    http://localhost:8080/produits
public class ProduitRESTController {
     @Autowired // pour l'injection de dépendances
     private ProduitRepository produitRepos;

     //  Message d'accueil
     //  http://localhost:8080/produits/index  (GET)
     @GetMapping(value ="/index" )
     public String accueil() {
          return "BienVenue au service Web REST 'produits'.....";
     }

     //Afficher la liste des produits
      //  http://localhost:8080/produits/ (GET)
          @GetMapping(
                       // spécifier le path de la méthode
                       value= "/")
          public  List<Produit> getAllProduits() {
               return produitRepos.findAll();
          }
}
```

23. Enregistrer, réexécuter et visualiser l'affichage des produits via le lien suivant :

http://localhost:8080/produits/

Remarquer un blocage lors de l'affichage dû à la relation bidirectionnelle entre l'entité « **Categorie** » et l'entité « **Produit** ». Ceci provoque des appels récursifs bloquant l'affichage.

24. Pour résoudre ce problème, aller à la classe «**Categorie** » pour indiquer à **spring** d'ignorer les appels aux produits en mode JSON : Ajouter l'annotation « **@JsonIgnore** » juste avant la définition de la méthode « **getProduits** » (Càd ignorer la liste des produits lors de la représentation JSON d'un objet «**Categorie**») :

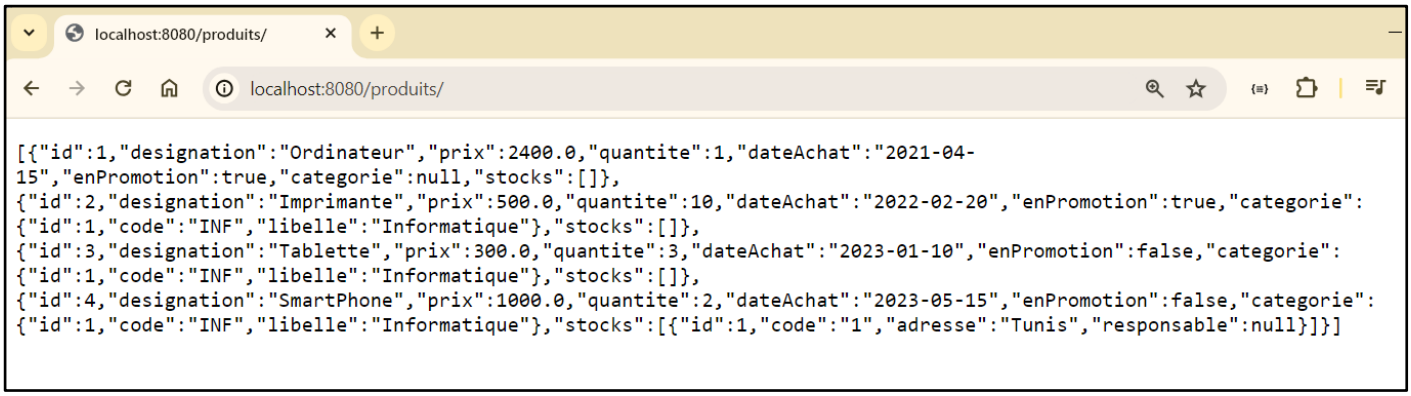
```
@JsonIgnore
public Collection<Produit> getProduits() {
     return produits;
}
```

NB : Ne pas oublier l'instruction de « **import** »

De même dans la classe « **Stock** » pour ignorer la liste des produits lors de la représentation JSON d'un objet « **Stock** »

25. Enregistrer la modification, réexécuter le projet et aller de nouveau au lien :

http://localhost:8080/produits/



26. Ajouter dans la classe «**ProduitRESTController**» une méthode « **getProduit** » qui permet de retourner un objet de type « Produit » ayant un « id » passé en paramètre :

- Elle porte le path « **/{id}** » (Le paramètre « **id** » du path est associé à l'argument « **id** » de la méthode en utilisant l'annotation « **@PathVariable** »)
- Utilise la méthode **GET** du http
- Utilise la méthode «**findById**» de l'interface «**ProduitRepository**» pour récupérer le produit donné.
- Et retourne les données sous le format **JSON** (par défaut)

Voici le code la méthode « **getProduit** » :

```
//  Afficher un produit en spécifiant son 'id'
//  http://localhost:8080/produits/{id} (GET)
@GetMapping(
            // spécifier le path de la méthode qui englobe un paramètre
            value= "/{id}"
            )
public Produit getProduit(@PathVariable Long id) {
      Produit p =produitRepos.findById(id).get();
      return p;
}
```

NB : importer le package « org.springframework.web.bind.annotation» pour l'annotation PathVariable

27. Enregistrer la modification, réexécuter le projet et visualiser l'affichage au format **JSON** (format par défaut) du produit ayant la valeur « **2** » pour l'attribut « **id** » à travers le lien suivant :

http://localhost:8080/produits/2

F. Afficher des données avec la méthode GET au format XML

28. Pour générer les représentations XML des données traitées par le service web REST, il est nécessaire d'ajouter la déclaration de la dépendance « **jackson-dataformat-xml** » dans le fichier « **pom.xml** ».Enregistrer le projet pour prendre en considération la nouvelle dépendance :

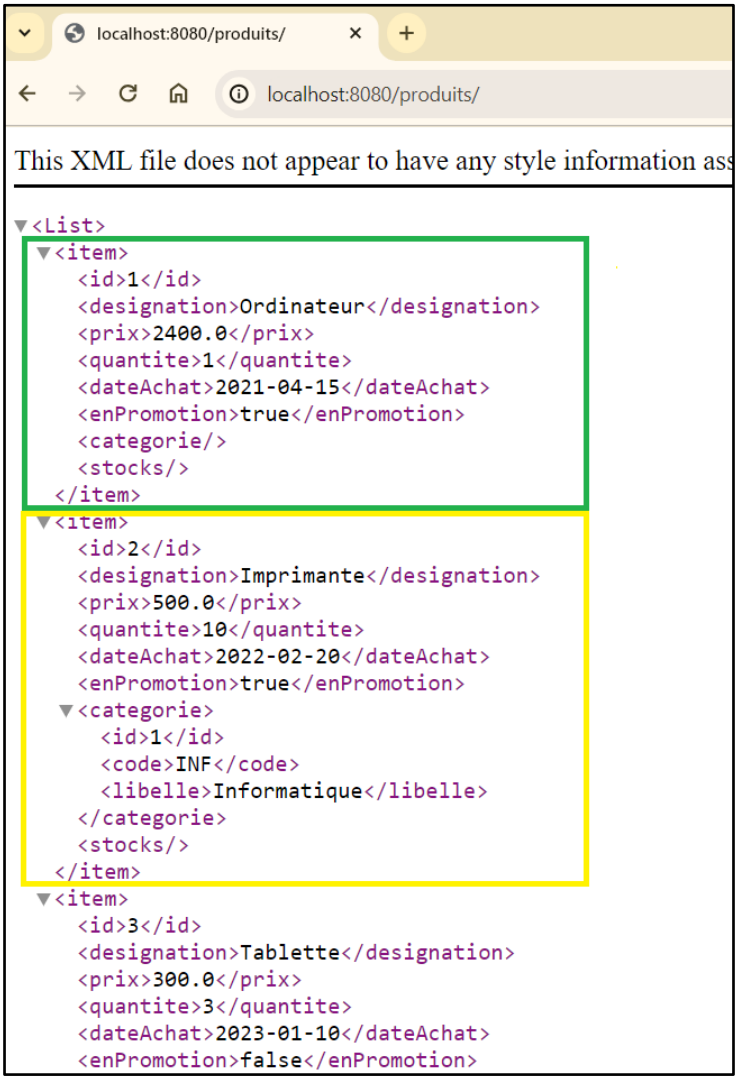
```
<dependency>
    <groupId>com.fasterxml.jackson.dataformat</groupId>
    <artifactId>jackson-dataformat-xml</artifactId>
</dependency>
```

29. Ensuite spécifier explicitement le format XML pour la représentation des données retournées par le service web (**produces = MediaType.APPLICATION_XML_VALUE**). Changer le code des deux méthodes « **getAllProduits** » et « **getProduit** » comme suit :

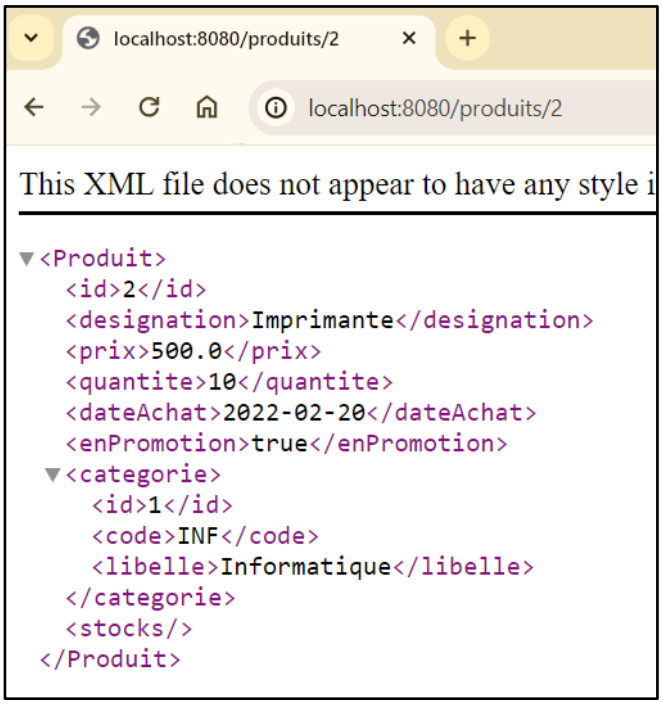
```
//  Afficher la liste des produits
//  http://localhost:8080/produits/ (GET)
        @GetMapping(
                        // spécifier le path de la méthode
                        value= "/",
                    // spécifier le format de retour en XML
                        produces = MediaType.APPLICATION_XML_VALUE
                   )
        public  List<Produit> getAllProduits() {
              return produitRepos.findAll();

        }

 //  Afficher un produit en spécifiant son 'id'
 //  http://localhost:8080/produits/{id} (GET)
        @GetMapping(
                        // spécifier le path de la méthode qui englobe un paramètre
                        value= "/{id}" ,
                    // spécifier le format de retour en XML
                        produces = MediaType.APPLICATION_XML_VALUE
                  )
        public Produit getProduit(@PathVariable Long id) {
              Produit p =produitRepos.findById(id).get();
              return p;
        }
```

NB : importer le package « org.springframework.http » pour la classe MediaType

30. Enregistrer la modification, réexécuter le projet et relancer l'affichage de tous les produits (cette fois au format XML) :



31. Afficher le produit ayant la valeur « **2** » pour l'attribut « **id** » (au format XML) :

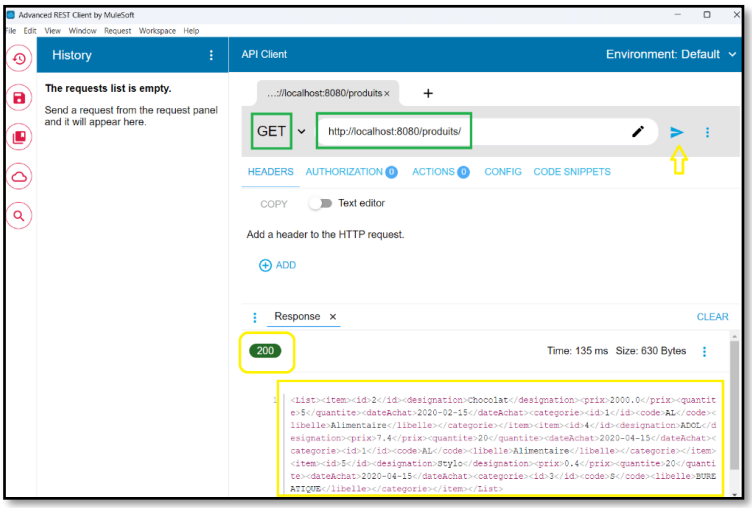


G. Test du service web REST à travers l'outil ARC du Google Chrome

Nous utilisons l'outil **RESTClient** (version Desktop) pour permettre de personnaliser les requêtes envoyées à un service RESTful. Il aide les programmeurs à développer l'application de test RESTful Service pour leurs services.

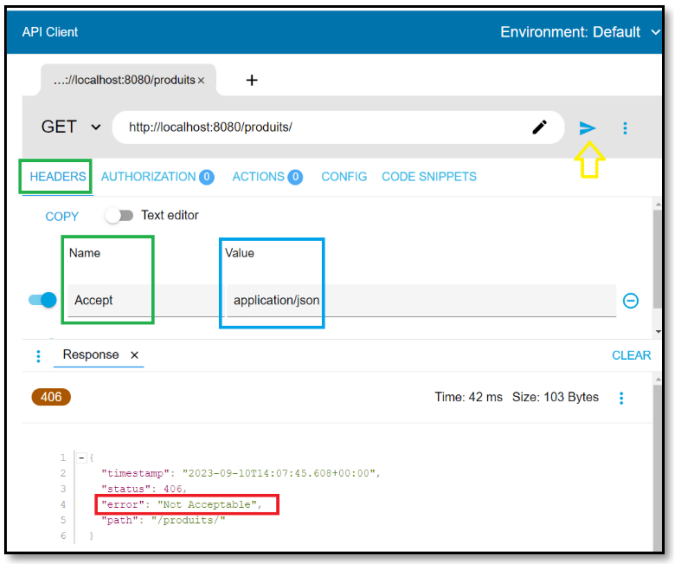
32. Passation de requête avec la méthode **GET**

- Spécifier le type de la méthode : **GET**
- Spécifier Le path de la requête : http://localhost:8080/produits/
- Pui envoyer la requête :



33. Dans la zone « **Headers** », ajouter une propriété à l'entête de la requête pour indiquer le format JSON pour la réponse:

- Header name = **Accept (format des données de la réponse)**
- Header value = **application/JSON**



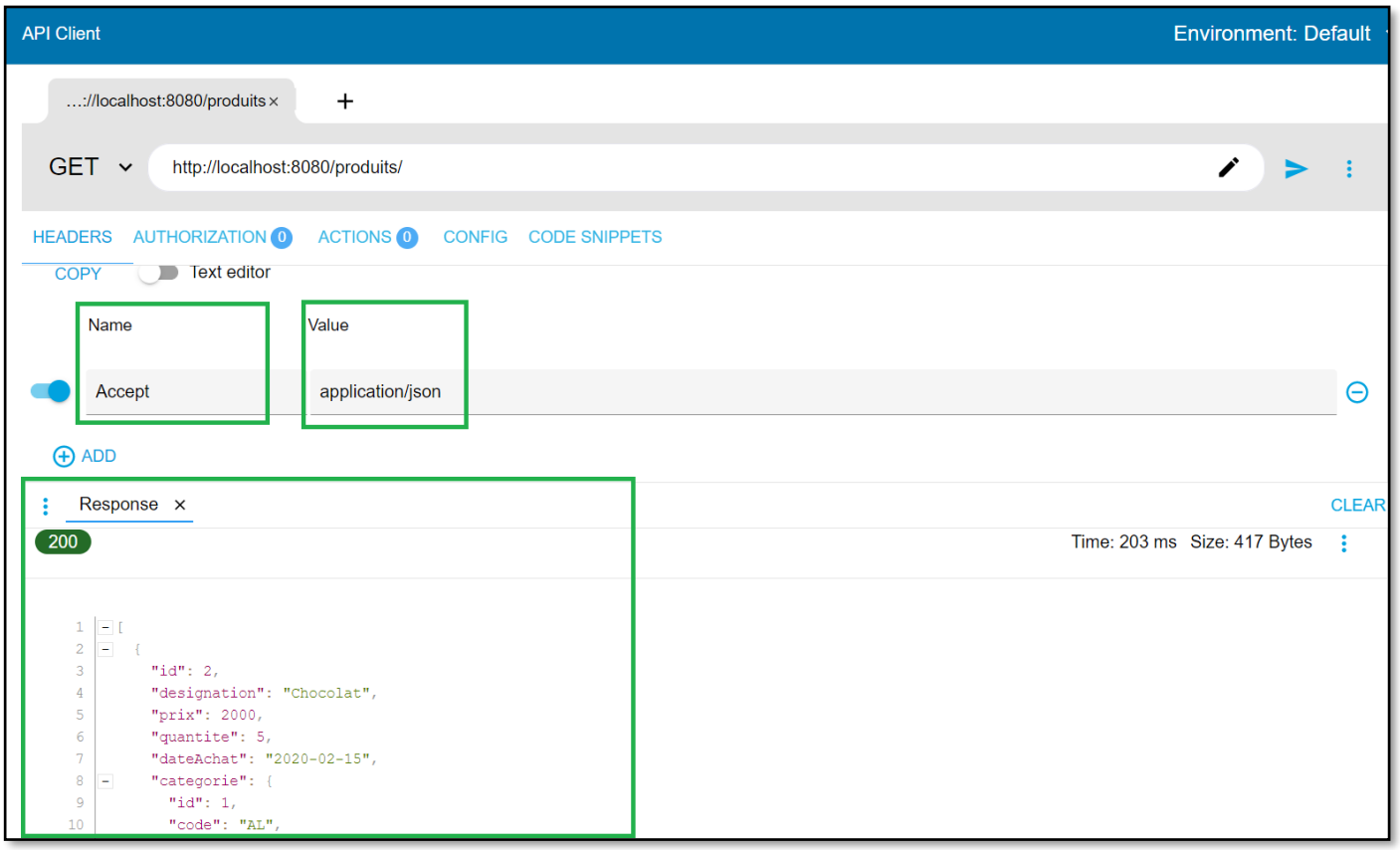
Le lancement de la requête déclenche une erreur 406 dans la réponse du serveur pour indiquer que le format JSON n'est pas supporté.

34. Pour avoir un affichage au format JSON :

- Ajouter, au niveau de l'annotation «**@GetMapping**» de la méthode « **getAllProduits**» le type du média JSON comme suit :

```
produces = {MediaType.APPLICATION_XML_VALUE , MediaType.APPLICATION_JSON_VALUE })
```

- Ainsi, le service web est capable d'envoyer des résultats avec les deux formats XML et JSON.
- Redémarrer Spring.
- Puis relancer la même requête en choisissant le format JSON (dans le header Accept) :



35. Réaliser les modifications nécessaires pour afficher, en utilisant l'outil ARC, le produit ayant la valeur « **2**» comme « **id** » au format XML puis au format **JSON :**

H. Supprimer un produit avec la méthode GET

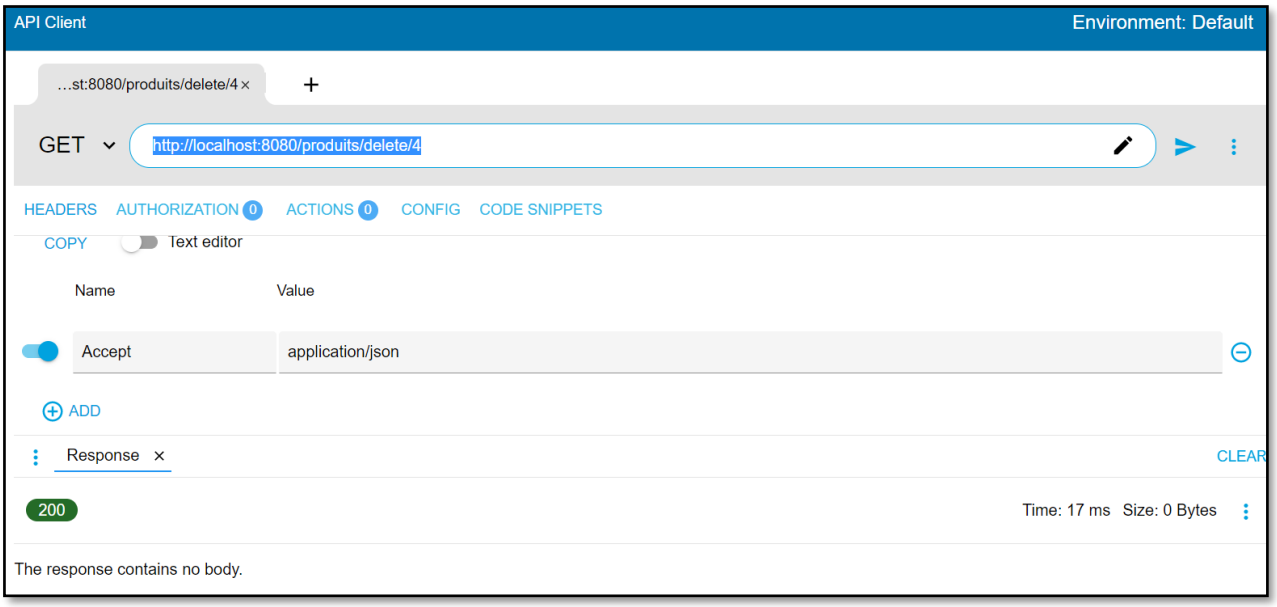
36. Ajouter dans la classe «**ProduitRESTController**» une méthode «**deleteProduit** » qui permet de supprimer un objet de type « **Produit** » ayant un « **id** » passé en paramètre :

- Elle porte le path « **/delete/{id}** » (Le paramètre « **id** » du path est associé à l'argument « **id** » de la méthode en utilisant l'annotation « **@PathVariable** »)
- Utilise la méthode **GET** du http
- Utilise la méthode «**deleteById**» de l'interface «**ProduitRepository**» pour supprimer le produit ayant l''attribut « id » donné en paramètre.
- Voici le code la méthode «**deleteProduit**» :

```
// Supprimer un produit par 'id' avec la méthode 'GET'
//  http://localhost:8080/produits/delete/{id}  (GET)
    @GetMapping(
              // spécifier le path de la méthode
              value = "/delete/{id}")
    public void deleteProduit(@PathVariable Long id)
        {
         produitRepos.deleteById(id);
        }
```

37. Enregistrer la modification et tester la méthode «**deleteProduit**» avec la méthode **GET** à travers le lien suivant (pour supprimer le produit ayant l'id « **4** »):

http://localhost:8080/produits/delete/4



I. Ajouter un nouveau produit en format JSON avec la méthode POST

38. Ajouter dans la classe «**ProduitRESTController**» une méthode «**addProduit**» qui permet d'ajouter un objet de type « **Produit** » passé en paramètre :

- Elle porte le path « **/** »
- Utilise la méthode **POST** du http
- Reçoit un argument de type «**Produit** » précédé de l'annotation «**@RequestBody**» :

```
//  ajouter un produit avec la méthode "POST"
//  http://localhost:8080/produits/   (POST)
        @PostMapping(
                    // spécifier le path de la méthode
                    value = "/"  ,
                    //spécifier le format de retour
                    produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE  }
                    )
        public Produit saveProduit(@RequestBody Produit p)
        {
             return produitRepos.save(p);
        }
```

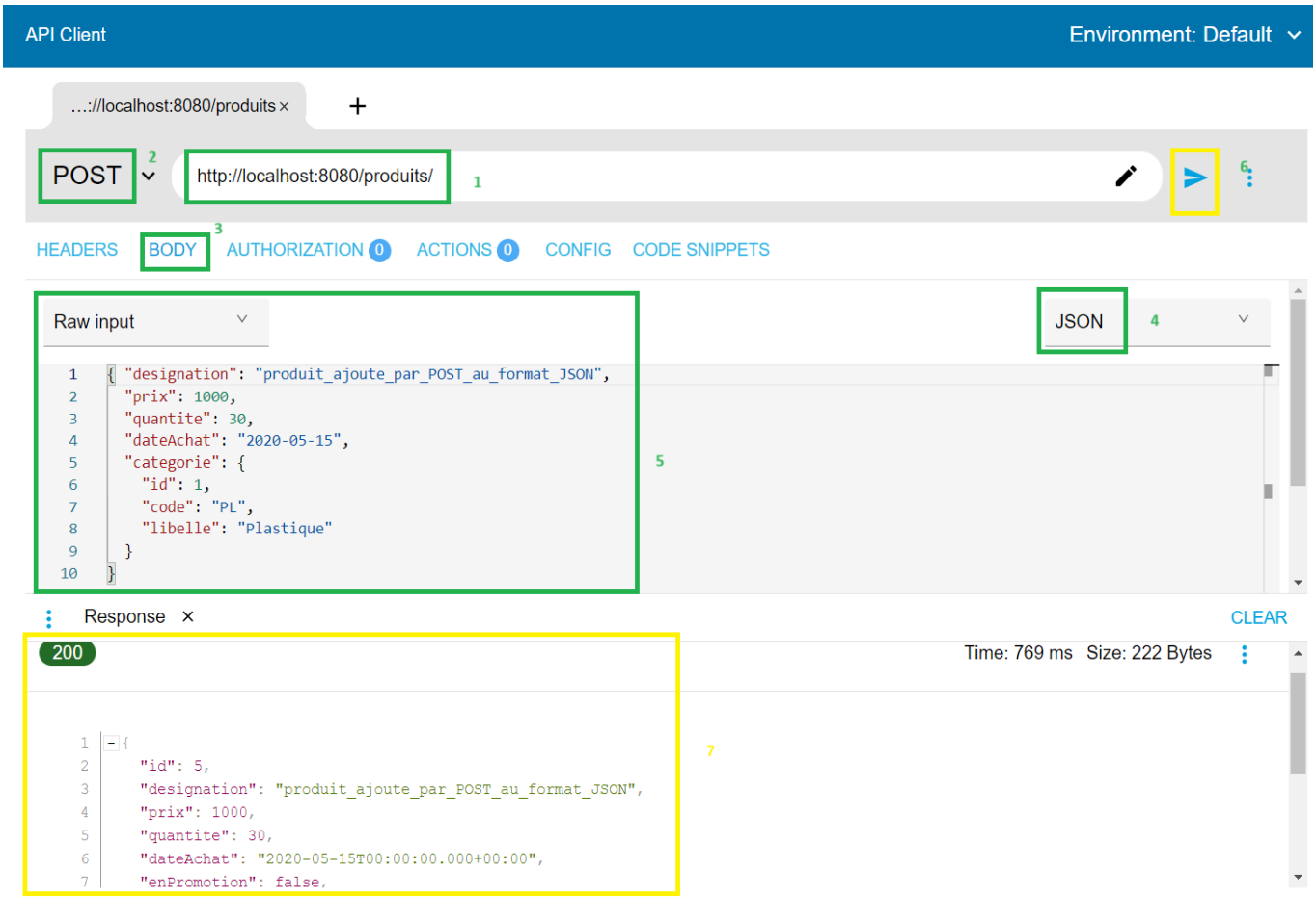
NB : importer le package « org.springframework.web.bind.annotation» pour la classe **RequestBody**

- Redémarrer Spring
- Au niveau de ARC , créer une nouvelle requête pour insérer le produit suivant au format JSON :

```
{ "designation": "produit_ajoute_par_POST_au_format_JSON",
  "prix": 1000,
  "quantite": 30,
  "dateAchat": "2020-05-15",
  "categorie": {
    "id": 1,
    "code": "PL",
    "libelle": "Plastique"
  }
}
```

- Spécifier l'URL suivante avec la méthode **POST** :

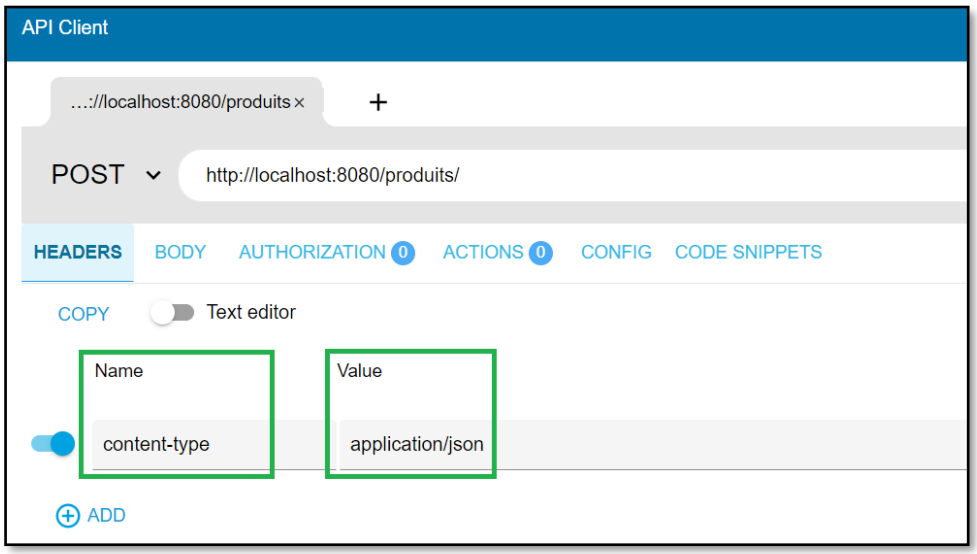
http://localhost:8080/produits/



39. Remarquer que l'outil ARC a ajouté un paramètre d'entête de la requête comme suit :

- Header name = **Content-Type (format des données de la requête)**
- Header value = **application/json**

Ceci est utile pour spécifier au serveur le format des données à recevoir :



40. Remarquer que la réponse est donnée avec la représentation **JSON** (par défaut) en associant au produit une valeur automatique à l'attribut « **id** ». Réaliser les configurations nécessaires pourque la réponse du serveur soit au format XML.

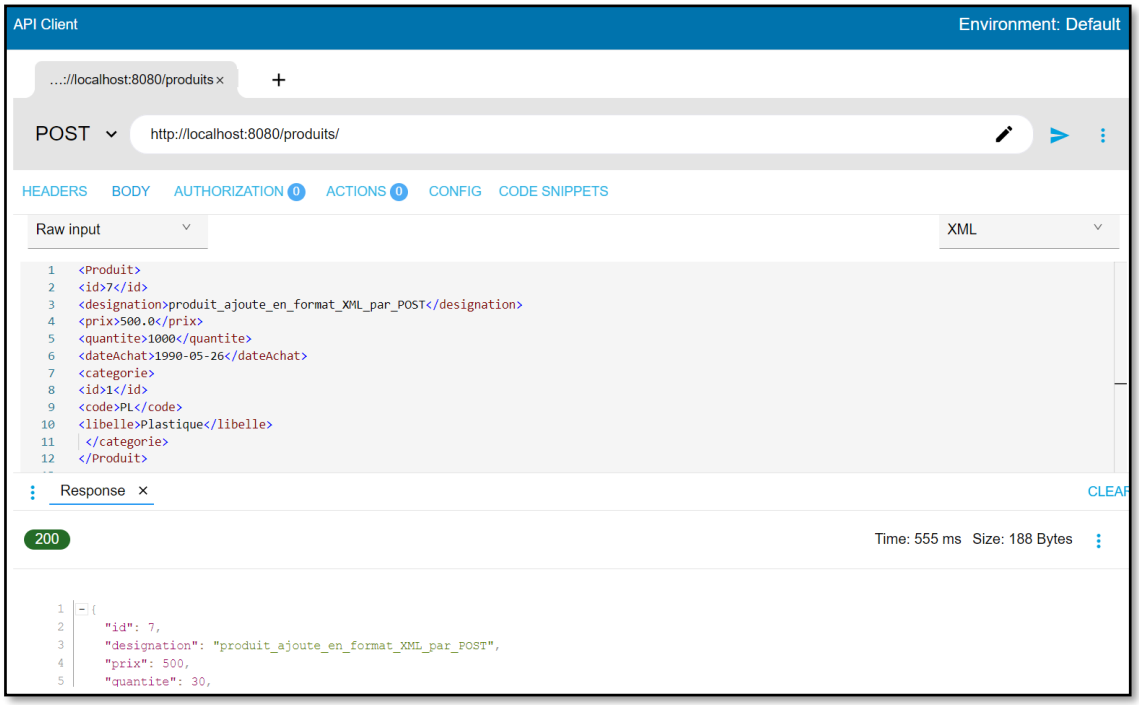
J. Ajouter un nouveau produit en format XML avec la méthode POST

41. Utiliser la même requête pour un nouveau produit en format XML :

- Spécifier la méthode POST
- Saisir l'URL
- Ajouter dans le volet « **Body** » un produit ayant les caractéristiques suivantes :

```
<Produit>
<designation>produit_ajoute_en_format_XML_par_POST</designation>
<prix>500</prix>
<quantite>30</quantite>
<dateAchat>1990-05-26</dateAchat>
<categorie>
<id>1</id>
<code>PL</code>
<libelle>Plastique</libelle>
 </categorie>
 </Produit>
```

- Envoyer la requête pour recevoir le résultat suivant :



42. Réaliser la modification nécessaire pour avoir une réponse en format JSON

K. Modifier un produit existant en format XML avec la méthode PUT

43. Ajouter dans la classe «**ProduitRESTController**» une méthode «**updateProduit**» qui permet de modifier un objet de type « **Produit** » passé en paramètre :

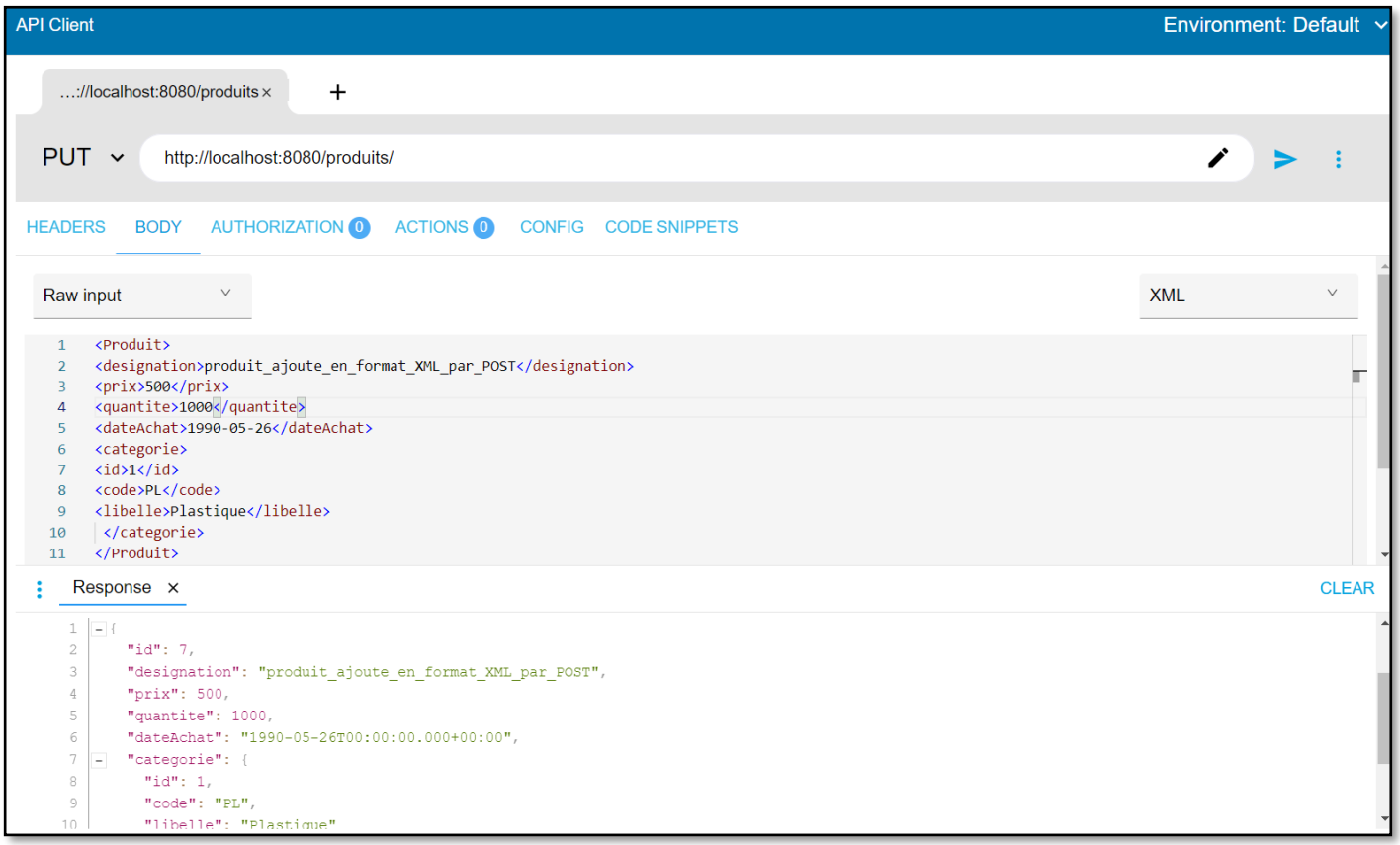
- Elle porte le path « **/** »
- Utilise la méthode **PUT** du http
- Reçoit un argument de type «**Produit** » précédé de l'annotation « **@RequestBody** »
- Utilise la méthode «**save**» de l'interface «**ProduitRepository**» :

```
//  modifier un produit avec la méthode "PUT"
//  http://localhost:8080/produits/   (PUT)
@PutMapping(
      // spécifier le path de la méthode
      value = "/"  ,
      //spécifier le format de retour
    produces = { MediaType.APPLICATION_JSON_VALUE, MediaType.APPLICATION_XML_VALUE  }
       )
      public Produit updateProduit(@RequestBody Produit p)
        {
            return produitRepos.save(p);
        }
```

- Enregistrer la modification et redémarrer Spring
- Tester la méthode «**updateProduit**» à travers l'outil ARC :
  - Utiliser la méthode PUT
  - Spécifier l'URL suivante :

    http://localhost:8080/produits/
  - Envoyer dans le body le produit suivant au format XML :

```
<Produit>
<id>7</id>
<designation>produit_ajoute_en_format_XML_par_POST</designation>
<prix>500.0</prix>
<quantite>100</quantite>
<dateAchat>1990-05-26</dateAchat>
<categorie>
<id>1</id>
<code>PL</code>
<libelle>Plastique</libelle>
 </categorie>
 </Produit>
```



44. Réaliser une modification d'un produit donné avec une représentation JSON.

L. Supprimer un produit existant en format XML avec la méthode DELETE

45. Ajouter dans la classe «**ProduitRESTController**» une méthode «**deleteProduit**» qui permet de supprimer un objet de type « **Produit** » passé en paramètre :

- Elle porte le path « **/** »
- Utilise la méthode **DELETE** du http
- Reçoit un argument de type «**Produit** » précédé de l'annotation « **@RequestBody** »

- Utilise la méthode «**delete**» de l'interface «**ProduitRepository**» pour supprimer le produit dernièrement créé en format XML :

```
<Produit>
<id>7</id>
<designation>produit_ajoute_en_format_XML_par_POST</designation>
<prix>500.0</prix>
<quantite>1000</quantite>
<dateAchat>1990-05-26</dateAchat>
<categorie>
<id>1</id>
<code>PL</code>
<libelle>Plastique</libelle>
 </categorie>
</Produit>
```

- Voici le code de la méthode « **deleteProduit** » :

```
// Supprimer un produit  avec la méthode 'DELETE'
//  http://localhost:8080/produits/   (DELETE)
@DeleteMapping(
// spécifier le path de la méthode
    value = "/")
public void deleteProduit(@RequestBody Produit p)
{
    produitRepos.delete(p);
}
```

- Enregistrer la modification et redémarrer Spring
- Passer au test avec ARC :