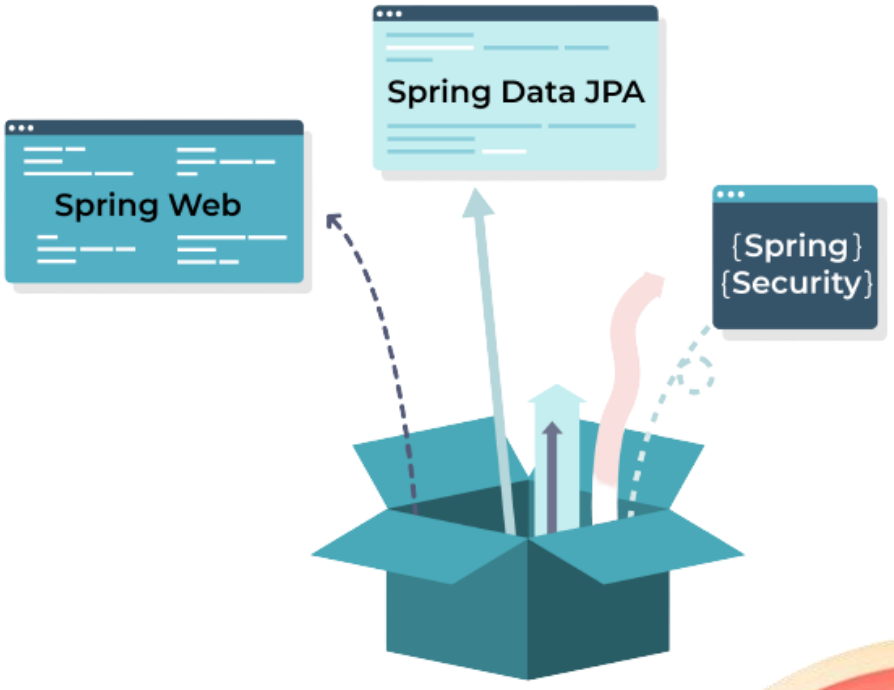
Compte rendu de :

SEANCE 09



Réalisé Par : AYOUJJIL Soukayna

Département : Mathématiques et Informatique

Filière : II-BDCC

Professeur : M. Mohamed YOUSSFI

Etape 1 : Création de l'entité Product

```java
package org.sid.ecomapp.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Product {
    @Id
    private String id;
    private  String name;
    private double price;
    private double quantity;

}
```

Etape 2 : Création de l'interface ProductRepository

```java
package org.sid.ecomapp.repositories;

import org.sid.ecomapp.entities.Product;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ProductRepository extends JpaRepository<Product,String> {
}
```

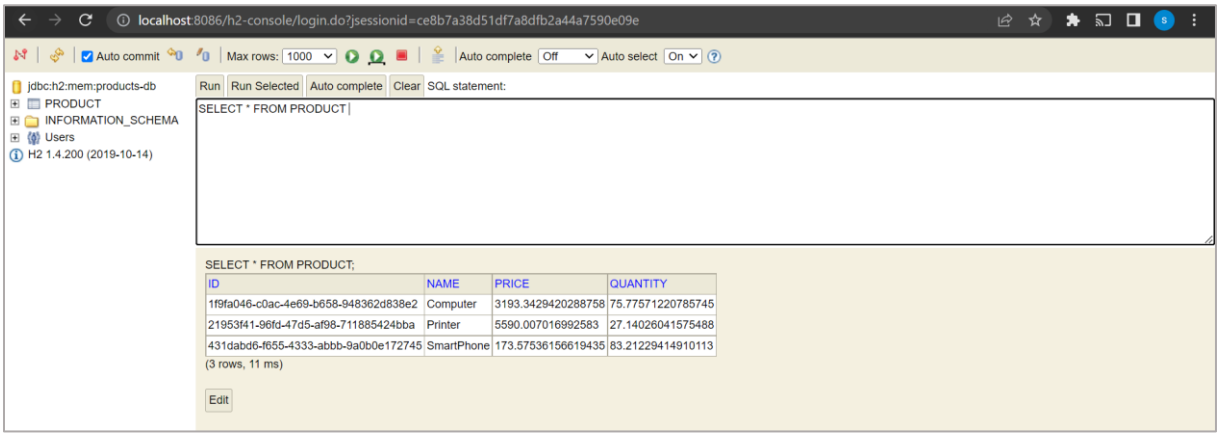
Etape 3 : Génération d'une liste des produits

```java
    @Bean
    public CommandLineRunner start(ProductRepository productRepository){
        return args -> {
            Stream.of("Computer","Printer","SmartPhone").forEach(
                name->{
                    productRepository.save(new Product(UUID.randomUUID().toString(),name, price: Math.random()*8000, quantity: Math.random()*100));
                }
            );
        };
    }
}
```

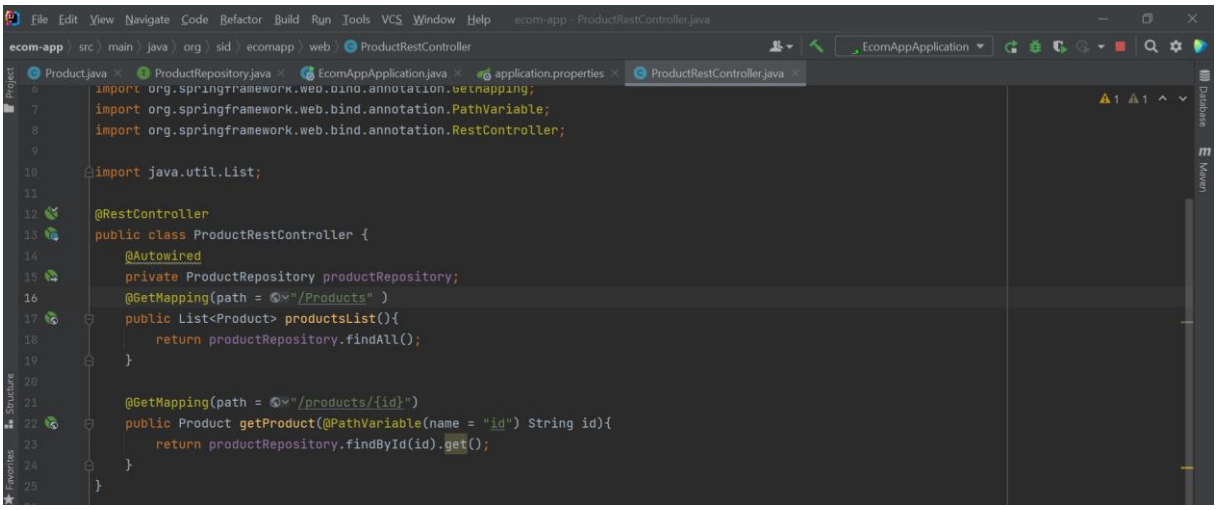
Etape 4 : Compléter le fichier application.properties

```
spring.datasource.url=jdbc:h2:mem:products-db
spring.h2.console.enabled=true
server.port=8086
```

Etape 5 : Démarrer l'application sur le port 8086



Etape 6 : créer un Rest Controller



```java
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class ProductRestController {
    @Autowired
    private ProductRepository productRepository;
    @GetMapping(path = "/Products" )
    public List<Product> productsList(){
        return productRepository.findAll();
    }

    @GetMapping(path = "/products/{id}")
    public Product getProduct(@PathVariable(name = "id") String id){
        return productRepository.findById(id).get();
    }
}
```

Etape 7 : Resultats

- Consulter Liste des produits

```
// 20220506152748
// http://localhost:8086/products/

[
  {
    "id": "d66ae583-7bce-4da2-b692-e1b6d67d7920",
    "name": "Computer",
    "price": 3830.281092201886,
    "quantity": 98.2543368738524
  },
  {
    "id": "7854d052-1825-4046-bf5f-e9a04980ce36",
    "name": "Printer",
    "price": 3779.4895794661948,
    "quantity": 91.79667108176004
  },
  {
    "id": "6073390d-48bf-40e4-9f68-e1cdc05834a5",
    "name": "SmartPhone",
    "price": 2979.9303094356064,
    "quantity": 18.00750706521238
  }
]
```

- Consulter un produit a l'aide de son id

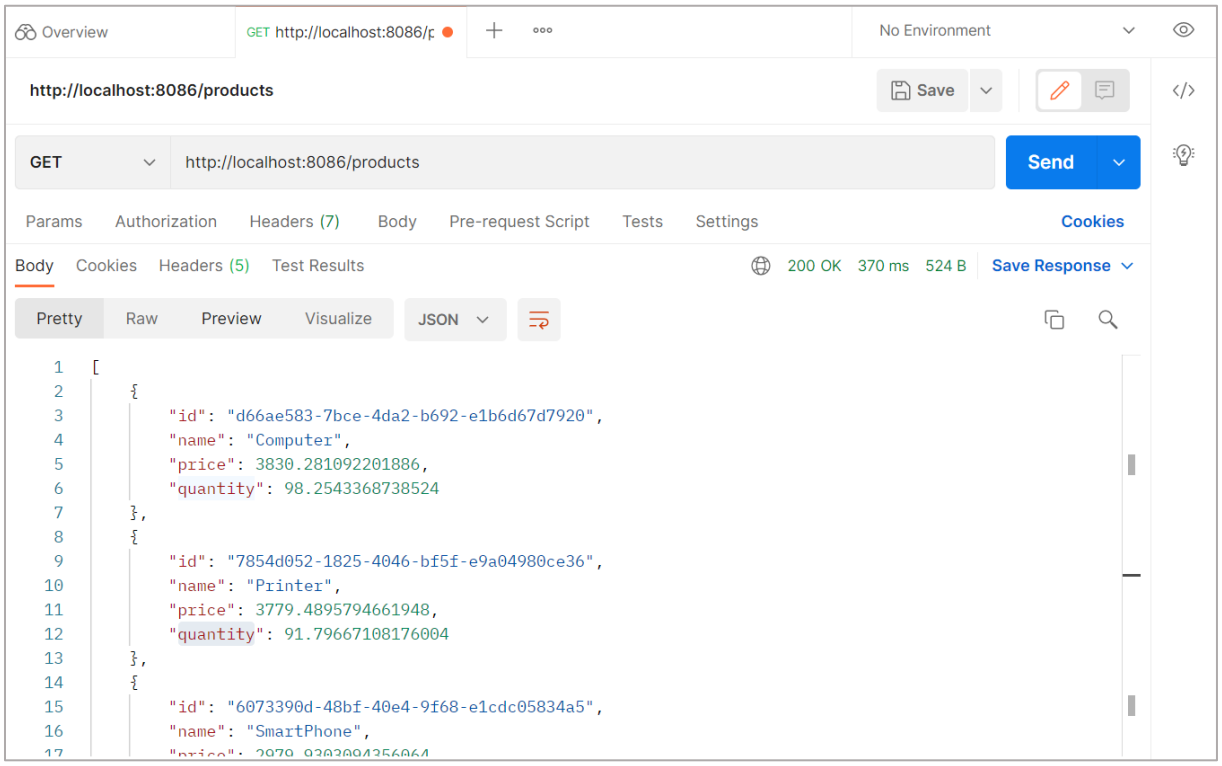
```
// 20220506151207
// http://localhost:8086/products/93ce5838-3a3b-4243-9228-78e31db86434

{
  "id": "93ce5838-3a3b-4243-9228-78e31db86434",
  "name": "Computer",
  "price": 3887.7695693915202,
  "quantity": 70.2633762342597
}
```

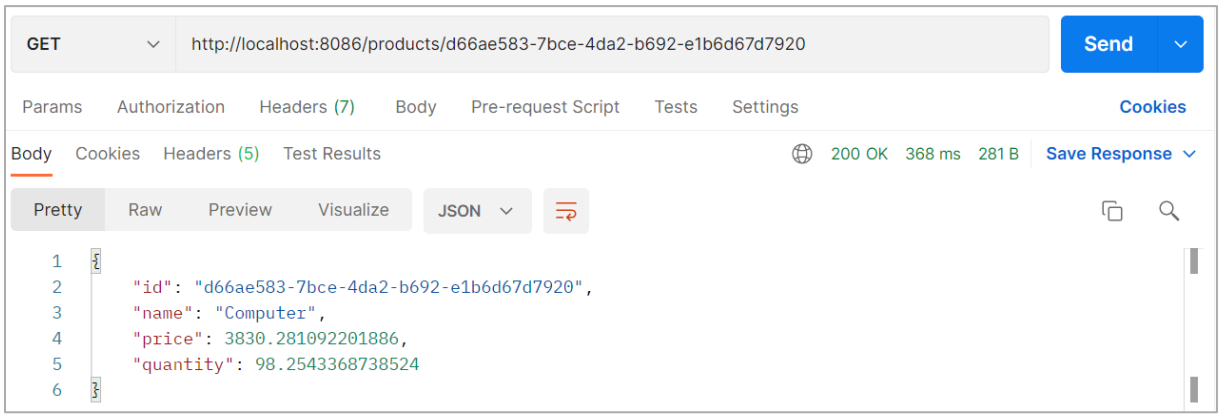
Etape 7 : Création des méthodes ajouter, modifier et supprimer un produit

```java
@PostMapping(path = "/products")
public Product saveProduct(@RequestBody Product p){
    p.setId(UUID.randomUUID().toString());
    return productRepository.save(p);
}

@PutMapping(path = "/products/{id}")
public Product updateProduct(@RequestBody Product p,@PathVariable(name = "id") String id){
    p.setId(id);
    return productRepository.save(p);
}

@DeleteMapping(path = "/products/{id}")
public void deleteProduct(@PathVariable(name = "id") String id){
    productRepository.deleteById(id);
}
```

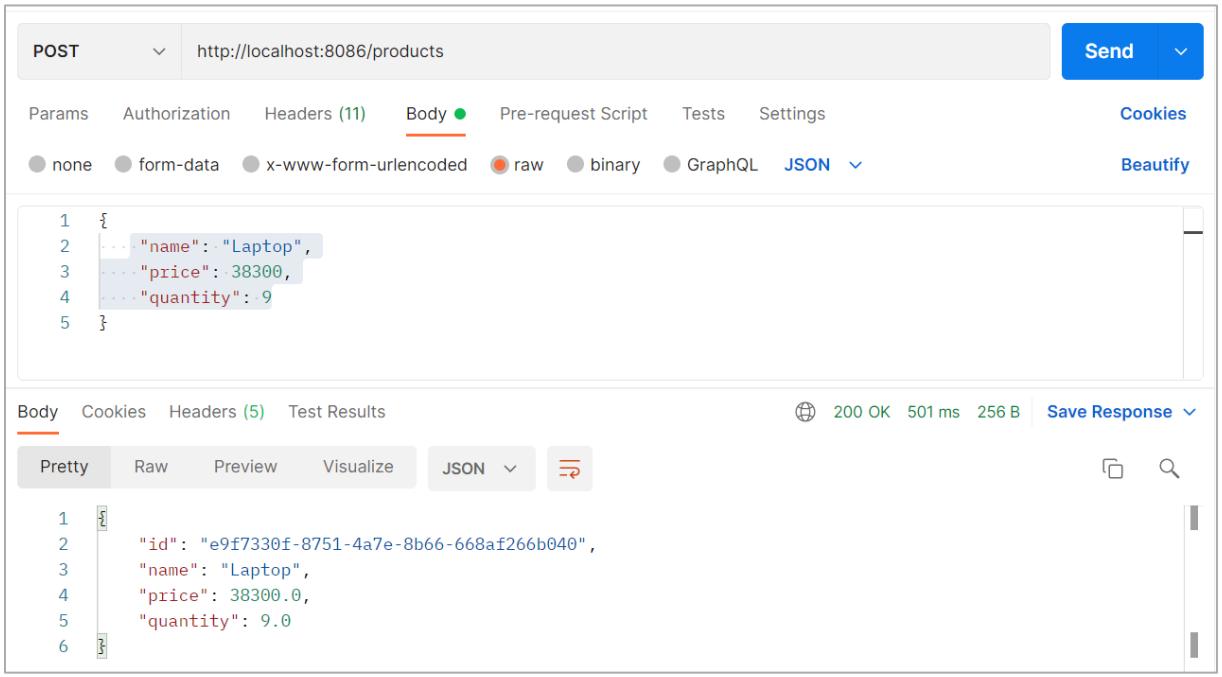
Etape 8 : Resultats

- Consulter Liste des produits / un seul produit

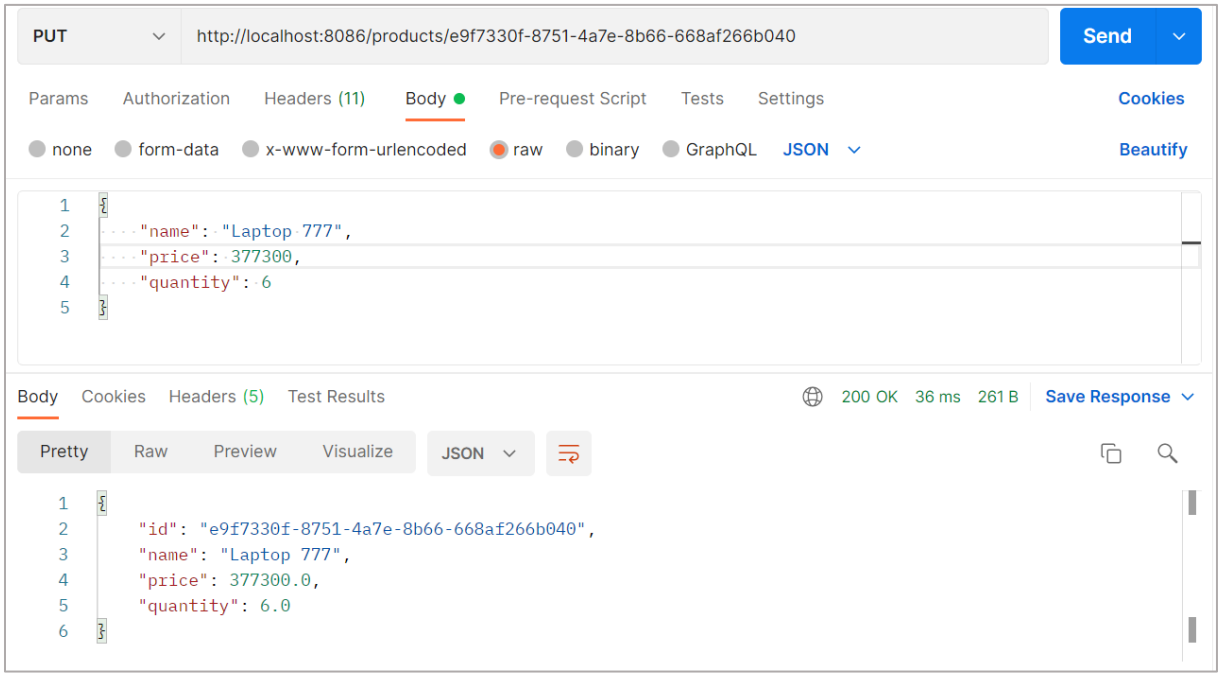




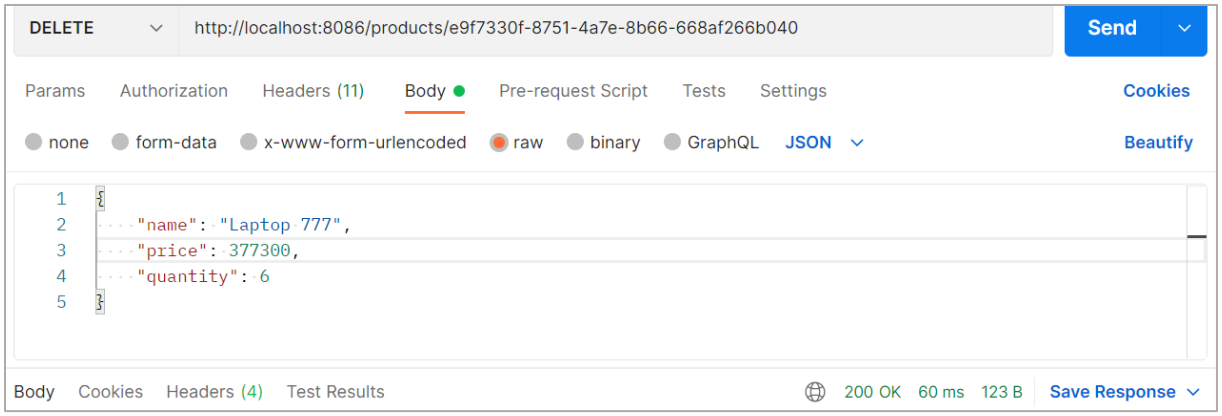
- Ajouter un produit



- Modifier un produit



- Supprimer un produit



Etape 9 : Création de l'entité Category

```java
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Category {
   @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @OneToMany(mappedBy = "category")
    private List<Product> products = new ArrayList<>();
}
```

Etape 10 : Création du repository CategoryRespository

```java
package org.sid.ecomapp.repositories;

import ...

public interface CategoryRepository extends JpaRepository<Category,Long> {
}
```

Etape 11 : Création des produits et categories

```java
@Bean
public CommandLineRunner start(ProductRepository productRepository, CategoryRepository categoryRepository){
    return args -> {
        Stream.of("Computer","Printer","SmartPhone").forEach(
                name->{
                    Category c = new Category();
                    c.setName(name);
                    categoryRepository.save(c);
                }
        );
        categoryRepository.findAll().forEach(cat->{
            for (int i = 1;i<=5;i++){
                Product product =  new Product();
                product.setId(UUID.randomUUID().toString());
                product.setPrice(Math.random()*9000);
                product.setQuantity(1+Math.random()*50);
                product.setName(cat.getName()+"-"+i);
                product.setCategory(cat);
                productRepository.save(product);

            }
        });
    };
```

Etape 12 : Resultats

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM CATEGORY
```

SELECT * FROM CATEGORY;

ID	NAME
1	Computer
2	Printer
3	SmartPhone

(3 rows, 5 ms)

Edit

Run | Run Selected | Auto complete | Clear | SQL statement:

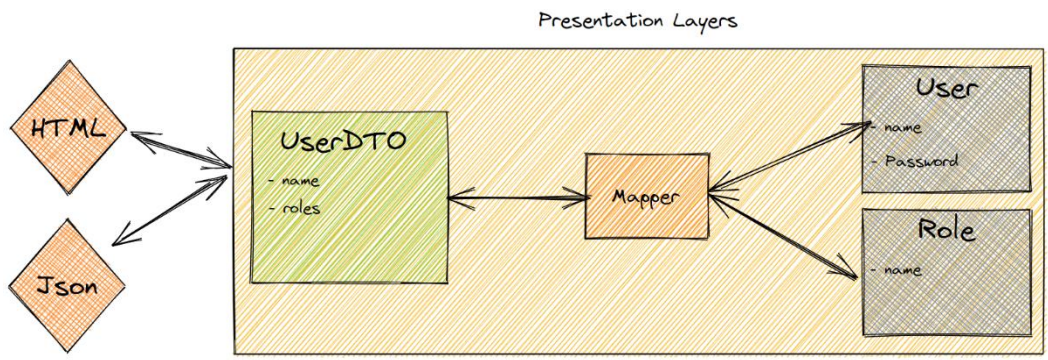
```
SELECT * FROM PRODUCT
```

SELECT * FROM PRODUCT;

ID	NAME	PRICE	QUANTITY	CATEGORY_ID
e11b32f7-4ae8-4ef0-b18e-7261c8f5ebd6	Computer-1	8690.369622665423	27.835907078367278	1
af40866e-bd04-4cef-b73f-bb7e862b59ab	Computer-2	3923.2621200083927	42.34470091642437	1
bdf93eb3-fa5c-45e1-8b34-94f89eb0ede4	Computer-3	2632.2677621007715	45.29745959716875	1
416f5e65-4261-4227-aac3-cb262a6024c7	Computer-4	6965.283708734053	48.706114414954065	1
26e46c78-382d-4c9d-a5db-927d03c49f8a	Computer-5	4139.458036526757	44.95812141822144	1
ea1d912c-21c1-4e1b-b74c-ed4191ca4305	Printer-1	4656.072134112546	34.02476313188289	2
094c0fe9-843c-4db4-8c06-f0bd8c440d7e	Printer-2	6346.829095895288	23.153481209347532	2
2d441081-62df-4250-93bc-24eb4848586d	Printer-3	1105.5035024424124	11.732040830526202	2
b2b0449b-770b-4c28-bcc8-11ba13c40dee	Printer-4	4138.005551834449	3.4543258990348926	2
404b3853-e51a-4252-bf7a-ec6b6377547e	Printer-5	6902.932038232382	18.461311529391104	2
bd91c073-7646-4f7d-97f8-f0e38ad9689b	SmartPhone-1	6731.794046346893	20.371802571967017	3
b5b1fbfb-818c-44f4-b272-88358274ff10	SmartPhone-2	172.0979852288874	44.683030156864625	3
51982b7d-b251-434e-b278-ed0b66f342b6	SmartPhone-3	2036.7356100998454	3.0691148643116772	3
e29807a7-617a-4127-b07e-26a5d69f5572	SmartPhone-4	7759.755813583914	11.825430665622854	3
03c785e9-c157-4565-8f6b-98196d5a3e7b	SmartPhone-5	8132.190886647338	15.819562097553673	3

Etape 13 : Création du modèle DTO (objet de transfert de données)

- **Comment l'utiliser ?**



https://www.baeldung.com/java-dto-pattern#how-to-use-it

- **Quand l'utiliser :**

Les DTO sont utiles dans les systèmes avec des appels à distance, car ils aident à en réduire le nombre.

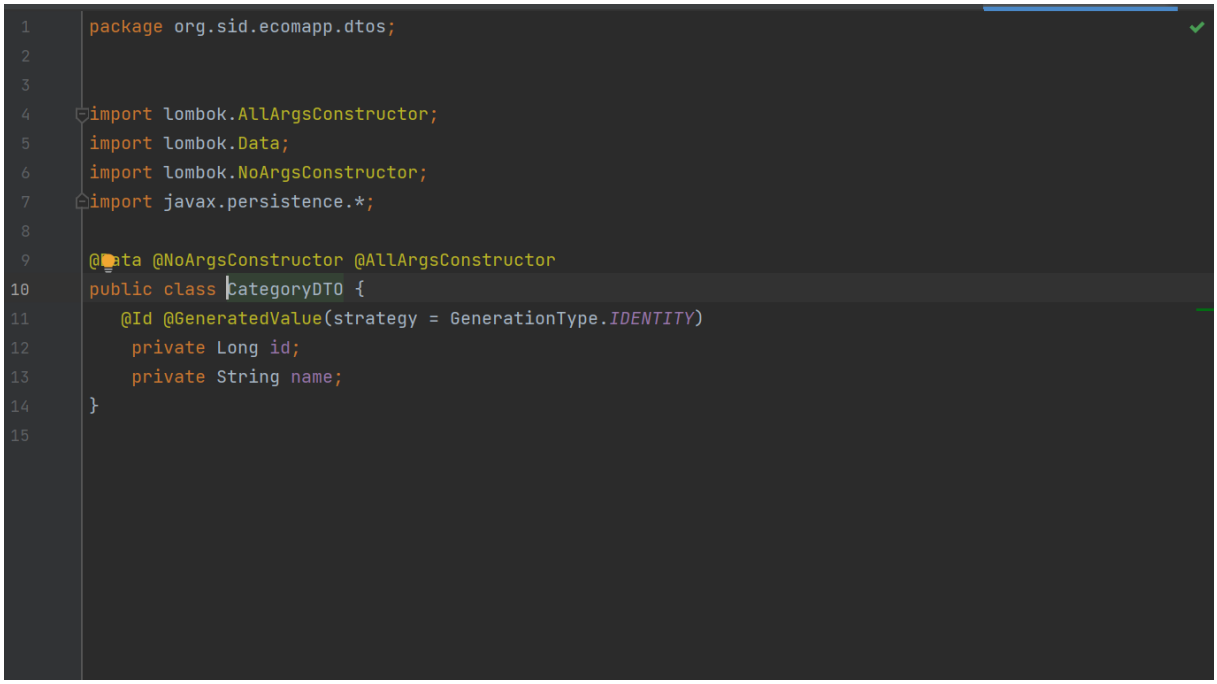
Les DTO sont également utiles lorsque le modèle de domaine est composé de nombreux objets différents et que le modèle de présentation a besoin de toutes leurs données en même temps, ou ils peuvent même réduire les allers-retours entre le client et le serveur.

Avec les DTO, nous pouvons créer différentes vues à partir de nos modèles de domaine, ce qui nous permet de créer d'autres représentations du même domaine mais en les optimisant en fonction des besoins des clients sans affecter notre conception de domaine. Une telle flexibilité est un outil puissant pour résoudre des problèmes complexes.

https://www.baeldung.com/java-dto-pattern#how-to-use-it

➢ Classe ProductDTO

➢ Classe CategoryDTO



Etape 14 : Création de la couche service : partie métier de l'application

- Interface IProductService

```java
public interface IProductService {
    ProductDTO save(ProductDTO productDTO);
    List<ProductDTO> productsList();
    ProductDTO getProduct(String id);
    ProductDTO updateProduct(ProductDTO productDTO);
    void deleteProduct(String id);
}
```

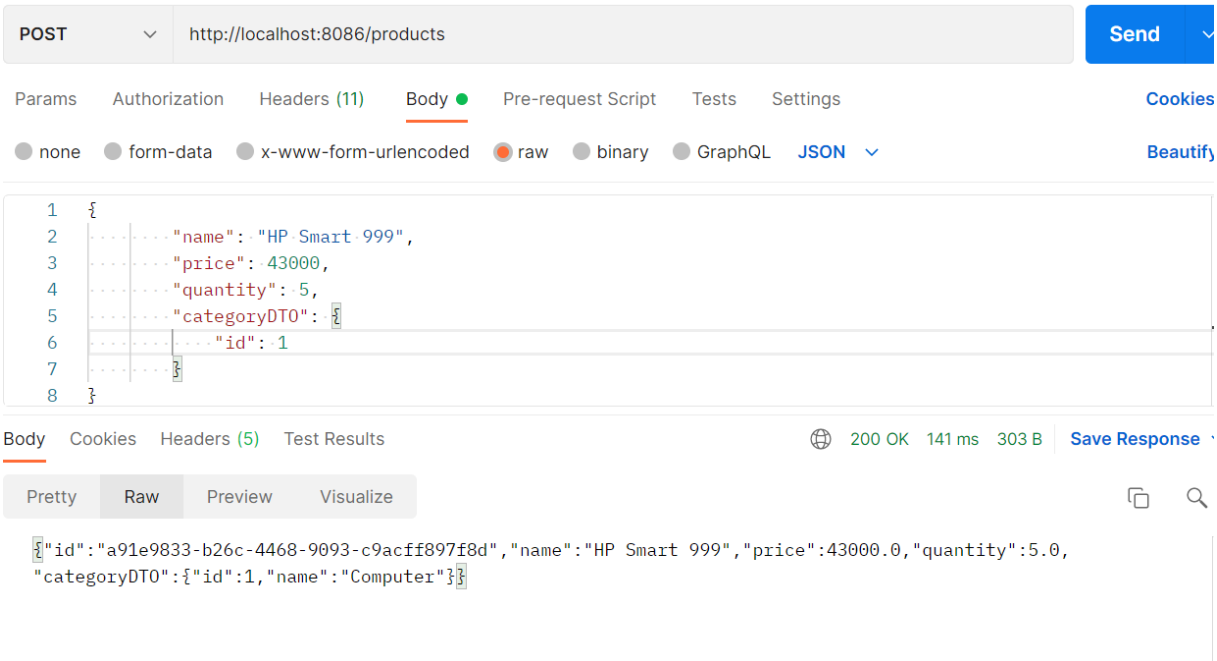
- Implémentation de l'interface IProductService

```java
@Service
@Transactional
public class IProductServiceImpl implements IProductService {
    @Autowired
    private ProductRepository productRepository;
    @Autowired
    private CatalogMappers catalogMappers;
    @Override
    public ProductDTO save(ProductDTO productDTO) {
       Product product =  catalogMappers.fromProductDTO(productDTO);
        product.setId(UUID.randomUUID().toString());
        Product savedProduct = productRepository.save(product);
       return catalogMappers.fromProduct(savedProduct);
    }

    @Override
    public List<ProductDTO> productsList() {
        List<Product> products = productRepository.findAll();
        List<ProductDTO> productDTOS = products.stream().map(
                p->catalogMappers.fromProduct(p)).collect(Collectors.toList());
        return productDTOS;
    }
}
```

```java
    @Autowired
    private IProductService iProductService;
    @GetMapping(path = "/products" )
    public List<ProductDTO> productsList(){
        return iProductService.productsList();
    }

    @GetMapping(path = "/products/{id}")
    public Product getProduct(@PathVariable(name = "id") String id){
        return productRepository.findById(id).get();
    }
    @PostMapping(path = "/products")
    public ProductDTO saveProduct(@RequestBody ProductDTO p){
        return iProductService.save(p);
    }
```

Etape 15 : Resultats

- Ajouter un produit

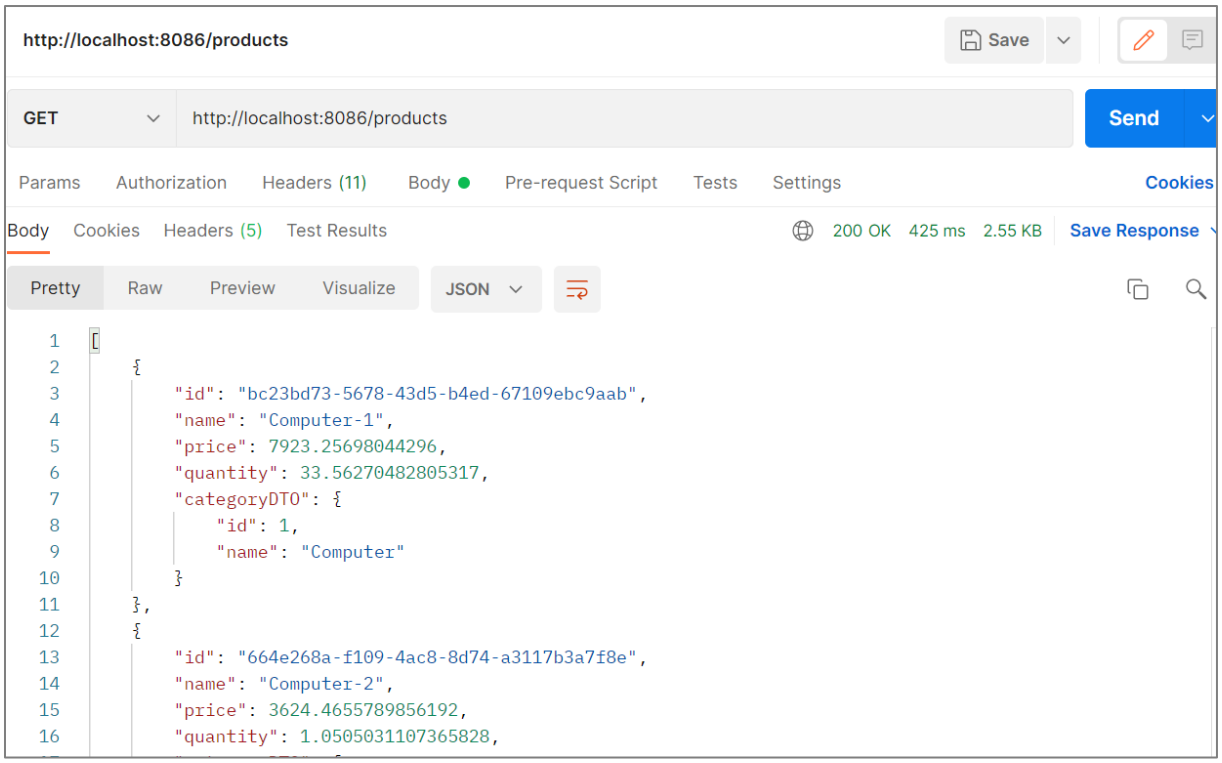


Run | Run Selected | Auto complete | Clear | SQL statement:

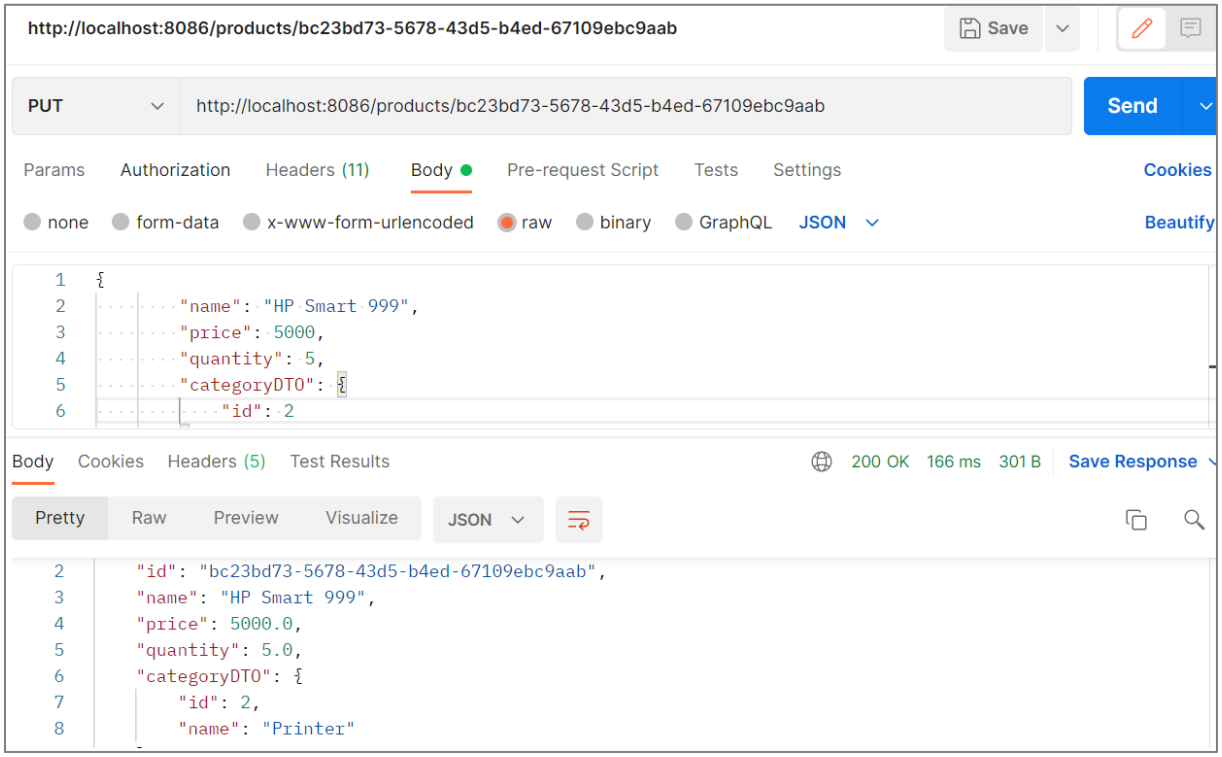
```
SELECT * FROM PRODUCT
```

71323eef-6bda-4615-be07-a6fc3ce33889	Computer-3	756.7807305058826	34.12235772857406	1
ed84824b-ba9e-4206-bf86-f5621c402bdb	Computer-4	2181.9143542519964	27.58128691062118	1
c584170f-90f0-4479-9da3-7812ea8d84a8	Computer-5	1539.190963231758	5.484767676228168	1
8f24ee59-6526-4ae8-bc9c-7c155be0b799	Printer-1	4554.990768911559	12.494807282089015	2
02143c22-b1ba-4724-b99d-a06d56db9b6f	Printer-2	7331.221654005137	44.505587273225636	2
ae0de449-9d67-4103-b7cf-f184c7462c3e	Printer-3	6764.580773640632	7.212428232712636	2
0b2f2280-f49b-4a02-bfe8-d656babb8c37	Printer-4	5022.935241185435	41.69267043276204	2
affd5d0c-9887-4c86-8de3-a0af6cec4cd1	Printer-5	1229.359694788638	31.960759432983863	2
51fbff9e-c8fe-48d2-be51-6d936626f77b	SmartPhone-1	6270.890678283063	41.45621041785973	3
92011903-7461-4d8c-8dd9-cf8c0e3b3c2f	SmartPhone-2	3295.994912324504	20.48399733193976	3
04eddfaa-3c47-4d20-b262-3b54cbf96c06	SmartPhone-3	8188.423743927855	24.254390898484843	3
af176e03-0ae2-4d5c-8951-35fdf26b7d7f	SmartPhone-4	2192.4204158242414	40.39646427636488	3
c3f44777-dfea-4edb-aa91-1fc3eb5edd6a	SmartPhone-5	2142.732841216112	36.71079815226233	3
a91e9833-b26c-4468-9093-c9acff897f8d	HP Smart 999	43000.0	5.0	1

(16 rows, 7 ms)



- Modifier un produit



SELECT * FROM PRODUCT;

ID	NAME	PRICE	QUANTITY	CATEGORY_ID
bc23bd73-5678-43d5-b4ed-67109ebc9aab	HP Smart 999	5000.0	5.0	2

- Supprimer un produit