Spring boot: le Design pattern DTO

(Data Transfer Object)

Objectifs :

1. Créer la classe *ProduitDTO*
2. Ajouter la méthode *convertEntityToDto()* à la couche *Service*
3. Modifier la classe *ProduitRESTController*
4. Ajouter la méthode *convertDtoToEntity()* à la couche *Service*
5. Utiliser la bibliothèque ***ModelMapper***

Créer la classe ProduitDTO

```
package com.nadhem.produits.dto;

import java.util.Date;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ProduitDTO {
	private Long idProduit;
	private String nomProduit;
	private Double prixProduit;
	private Date dateCreation;
	private Categorie categorie;
}
```

Ajouter la méthode convertEntityToDto() à la couche Service

Ajouter la méthode convertEntityToDto() à l'interface ProduitService

```
ProduitDTO convertEntityToDto (Produit produit);
```

Ajouter la méthode *convertEntityToDto()* à la classe ProdtuitServiceIml

```
@Override
public ProduitDTO convertEntityToDto(Produit produit) {
         ProduitDTO produitDTO = new ProduitDTO();
         produitDTO.setIdProduit(produit.getIdProduit());
         produitDTO.setNomProduit(produit.getNomProduit());
         produitDTO.setPrixProduit(produit.getPrixProduit());
        produitDTO.setDateCreation(p.getDateCreation());
         produitDTO.setCategorie(produit.getCategorie());
              return produitDTO;

         /*return ProduitDTO.builder()
                          .idProduit(produit.getIdProduit())
                          .nomProduit(produit.getNomProduit())
                          .prixProduit(produit.getPrixProduit())
                          .dateCreation(p.getDateCreation())
                          .categorie(produit.getCategorie())
                          .build();*/
       }
```

Modifier les méthodes de ProduitService pour retourner des DTO

```
public interface ProduitService {

         ProduitDTO saveProduit(Produit p);
         ProduitDTO getProduit(Long id);
         List<ProduitDTO> getAllProduits();

…
```

```
@Service
public class ProduitServiceImpl implements ProduitService {

       @Autowired
       ProduitRepository produitRepository;

       @Override
       public ProduitDTO saveProduit(Produit p) {
              return  convertEntityToDto( produitRepository.save(p));
       }

       @Override
       public ProduitDTO getProduit(Long id) {
              return  convertEntityToDto( produitRepository.findById(id).get());

       }

       @Override
       public List<ProduitDTO> getAllProduits() {
```

```
		return produitRepository.findAll().stream()
				.map(this::convertEntityToDto)
				.collect(Collectors.toList());

		//OU BIEN
		/*List<Produit> prods =  produitRepository.findAll();
		List<ProduitDTO> listprodDto = new ArrayList<>(prods.size());
		for (Produit p : prods)
			listprodDto.add(convertEntityToDto(p));
		return listprodDto;*/
	}
```

Modifier la classe *ProduitRESTController*

```
public class ProduitRESTController {
@Autowired
	ProduitService produitService;

	@RequestMapping(method = RequestMethod.GET)
	public List<ProduitDTO> getAllProduits() {
		return produitService.getAllProduits();
	 }

	@RequestMapping(value="/{id}",method = RequestMethod.GET)
	public ProduitDTO getProduitById(@PathVariable("id") Long id) {
		return produitService.getProduit(id);
    }
```

…

Tester avec PostMan l'api getAllProduits

Masquer l'attribut prixProduit

Ajouter l'attribut nomCategorie à la classe ProduitDTO

```
	private String nomCat;
```

Ajouter la méthode *convertDtoToEntity()* au service

```
Produit convertDtoToEntity(ProduitDTO produitDto);

@Override
public Produit convertDtoToEntity(ProduitDTO produitDto) {
		Produit produit = new Produit();
		produit.setIdProduit(produitDto.getIdProduit());
		produit.setNomProduit(produitDto.getNomProduit());
		produit.setPrixProduit(produitDto.getPrixProduit());
		produit.setDateCreation(produitDto.getDateCreation());
		produit.setCategorie(produitDto.getCategorie());
			return produit;
	}
```

Modifier les méthodes *saveProduit ()* et *updateProduit ()* de la couche *Service*

```
ProduitDTO saveProduit(ProduitDTO p);
ProduitDTO updateProduit(ProduitDTO p);
```

```
@Override
public ProduitDTO saveProduit(ProduitDTO p) {
   return  convertEntityToDto( produitRepository.save(convertDtoToEntity(p)));
	}

@Override
public ProduitDTO updateProduit(ProduitDTO p) {
   return convertEntityToDto(produitRepository.save(convertDtoToEntity(p)))
	}
```

Modifier les méthodes *createProduit()* et *updateProduit()* de la classe *ProduitRESTController*

```
@RequestMapping(method = RequestMethod.POST)
public ProduitDTO createProduit(@RequestBody ProduitDTO produitDTO) {
		return produitService.saveProduit(produitDTO);
	}

@RequestMapping(method = RequestMethod.PUT)
public ProduitDTO updateProduit(@RequestBody ProduitDTO produitDTO) {
		return produitService.updateProduit(produitDTO);
	}
```

Utiliser la bibliothèque *ModelMapper*

Ajouter la dépendance

```
<dependency>
  <groupId>org.modelmapper</groupId>
  <artifactId>modelmapper</artifactId>
  <version>3.0.0</version>
</dependency>
```

Créer la bean dans la classe *ProduitsApplication*

```
@Bean
public ModelMapper modelMapper()
	{
		return new ModelMapper();
	}
```

Modifier la méthode convertEntityToDto() de la classe *ProduitServiceImpl*

```
@Autowired
	ModelMapper modelMapper;


@Override
public ProduitDTO convertEntityToDto(Produit produit) {
	ProduitDTO produitDTO = modelMapper.map(produit, ProduitDTO.class);
		return produitDTO;
		}
```

Tester avec POSTMAN

Ajouter l'attribut nomCat à la classe ProduitDTO

```
	private String nomCat;
```

Tester avec POSTMAN

```
	"nomCat": null
```

Modifier la méthode convertEntityToDto()

```
@Override
public ProduitDTO convertEntityToDto(Produit produit) {

modelMapper.getConfiguration().setMatchingStrategy(MatchingStrategies.LOOSE);
ProduitDTO produitDTO = modelMapper.map(produit, ProduitDTO.class);

…
```

Tester avec POSTMAN

```
 "nomCat": "cat2"
```

Modifier la méthode convertDtoToEntity() de la classe *ProduitServiceImpl*

```
@Override
public Produit convertDtoToEntity(ProduitDTO produitDto) {
            Produit produit = new Produit();
            produit = modelMapper.map(produitDto, Produit.class);
…
```

Tester l'ajout d'un nouveau produit avec POSTMAN