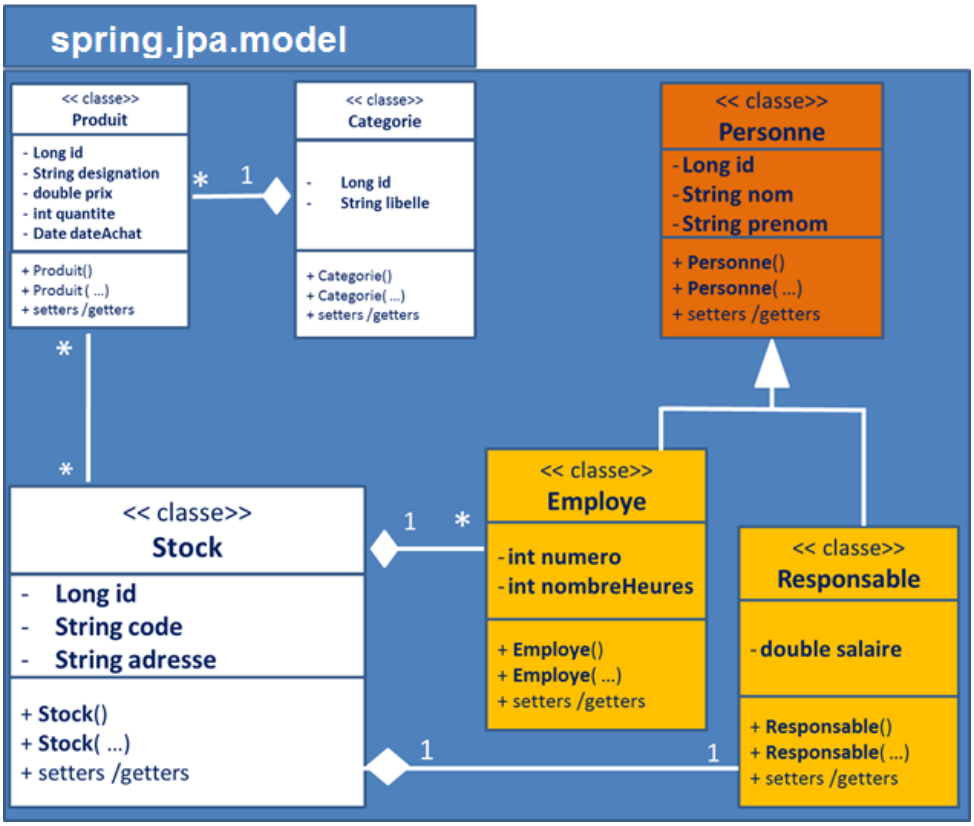
Atelier Spring -03-
JPA –Partie09

Objectifs

- **JPA et Spring Data**
  - Mettre en œuvre d'une relation d'héritage JPA
  - Stratégie : **MappedSuperclass**
  - Réaliser les requêtes polymorphes

Voici un modèle conceptuel qui modélise une relation d'héritage **:**

- Une sous-classe « **Responsable** » caractérisée par un **salaire**
- Une sous-classe « **Employe** » caractérisée par deux attributs (**numero** et **nombreHeures**)
- Une classe mère « **Personne** » qui présente les attributs communs (**id**, **nom** et **prenom**)



• La stratégie « MappedSuperclass »

1. Prendre une copie du projet «**jpa-spring-data-spring-boot-2-relations-3**» et le nommer «**jpa-spring-data-spring-boot-2-relations-3-heritage1».**
2. Créer une nouvelle classe «**Personne**» (classe mère) ayant le code suivant :

```
package spring.jpa.model;

import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.MappedSuperclass;
import jakarta.persistence.Id;

// pour déclarer une classe mère (non entité)
@MappedSuperclass
public class Personne {

    @Id
    @GeneratedValue (strategy = GenerationType.AUTO)
    protected  Long id;
    protected  String nom;
    protected  String prenom;

    public Personne(String nom, String prenom) {
           super();
           this.nom = nom;
           this.prenom = prenom;
     }
     public Long getId() {
           return id;
     }
     public void setId(Long id) {
           this.id = id;
     }
     public String getNom() {
           return nom;
     }
     public void setNom(String nom) {
           this.nom = nom;
     }
     public String getPrenom() {
           return prenom;
     }
     public void setPrenom(String prenom) {
           this.prenom = prenom;
     }
     public Personne() {
           super();
     }    }
```

3. Créer une nouvelle entité «**Employe**» (sous-classe) ayant le code suivant :

```java
package spring.jpa.model;

import jakarta.persistence.Entity;
import jakarta.persistence.ManyToOne;

@Entity
public class Employe extends Personne
{

   private int numero;
   private int nombreHeures;

      @ManyToOne
      private Stock stock;


      public Employe() {
            super();
            // TODO Auto-generated constructor stub
      }

public Employe(String nom, String prenom, int numero, int nombreHeures, Stock stock) {
            super(nom, prenom);
            this.numero = numero;
            this.nombreHeures = nombreHeures;
            this.stock = stock;
      }

      public Employe(String nom, String prenom, int numero, int nombreHeures) {
            super(nom, prenom);
            this.numero = numero;
            this.nombreHeures = nombreHeures;
      }


      public Employe(String nom, String prenom) {
            super(nom, prenom);
            // TODO Auto-generated constructor stub
      }

      public Employe(String nom, String prenom, int numero) {
            super(nom, prenom);
            this.numero = numero;
      }

      public int getNombreHeures() {
            return nombreHeures;
      }

      public void setNombreHeures(int nombreHeures) {
            this.nombreHeures = nombreHeures;
      }
      @Override
```

```
	public String toString() {
		return "Employe [numero=" + numero + ", nombreHeures=" + nombreHeures +
", stock=" + stock + "]";
	}

	public int getNumero() {
		return numero;
	}

	public void setNumero(int numero) {
		this.numero = numero;
	}

	public Stock getStock() {
		return stock;
	}

	public void setStock(Stock stock) {
		this.stock = stock;
	}
}
```

4. Remplacer le code de l'entité «**Responsable**» (sous-classe) par le code suivant :

```
package spring.jpa.model;

import jakarta.persistence.Entity;
import jakarta.persistence.OneToOne;

@Entity
public class Responsable  extends  Personne{

	private double salaire;

	@OneToOne (mappedBy = "responsable")
	private Stock stock;

	public Responsable(String nom, String prenom, double salaire, Stock stock) {
		super(nom, prenom);
		this.salaire = salaire;
		this.stock = stock;
	}

	public Responsable(String nom, String prenom, double salaire) {
		super(nom, prenom);
		this.salaire = salaire;
	}
	public Responsable(String nom, String prenom) {
		super(nom, prenom);
		// TODO Auto-generated constructor stub
	}
	public Responsable() {
```

```
            super();
            // TODO Auto-generated constructor stub
      }
      public String getNom() {
            return nom;
      }
      @Override
      public String toString() {
            return "Responsable [salaire=" + salaire + ", stock=" + stock + "]";
      }

      public double getSalaire() {
            return salaire;
      }
      public void setSalaire(double salaire) {
            this.salaire = salaire;
      }
      public void setNom(String nom) {
            this.nom = nom;
      }
      public String getPrenom() {
            return prenom;
      }

      public void setPrenom(String prenom) {
            this.prenom = prenom;
      }

      public Stock getStock() {
            return stock;
      }

      public void setStock(Stock stock) {
            this.stock = stock;
      }
}
```

5. Remplacer le code de l'entité «**Stock**» (pour ajouter une relation 1-N avec l'entité « **Employe** ») par le code suivant :

```
package spring.jpa.model;

import java.util.ArrayList;
import java.util.Collection;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
```

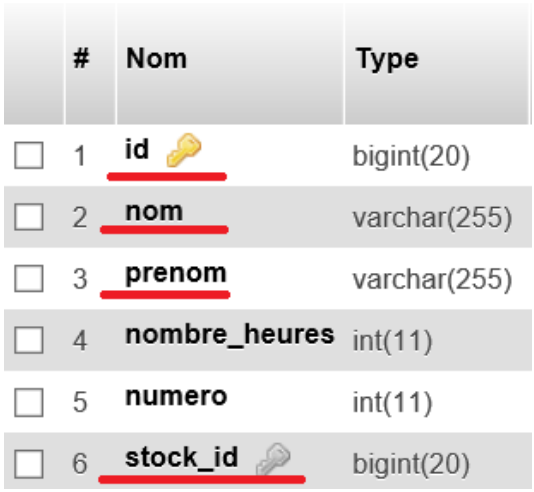
```java
@Entity
public class Stock
{
	@Id
	@GeneratedValue
	private Long id;

	@Column(length = 50)
	private String code;

	@Column(length = 50)
	private String adresse;

	@ManyToMany (mappedBy = "stocks", cascade = CascadeType.REMOVE)
	private Collection<Produit> produits = new ArrayList<Produit>();

	@OneToOne (cascade= {CascadeType.PERSIST})
	private Responsable responsable;


	@OneToMany (mappedBy = "stock" ,cascade = {CascadeType.REMOVE,
CascadeType.MERGE, CascadeType.PERSIST} )
    private Collection <Employe> employes = new
ArrayList<Employe>();


	public Collection<Employe> getEmployes() {
		return employes;
	}
	public void setEmployes(Collection<Employe> employes) {
		this.employes = employes;
	}
	public String getCode() {
		return code;
	}
	public Responsable getResponsable() {
		return responsable;
	}
	public void setResponsable(Responsable responsable) {
		this.responsable = responsable;
	}
	public Stock(String code, String adresse, Responsable responsable) {
		super();
		this.code = code;
		this.adresse = adresse;
		this.responsable = responsable;
	}
	public void setCode(String code) {
		this.code = code;
	}
```

```
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public Collection<Produit> getProduits() {
		return produits;
	}
	public void setProduits(Collection<Produit> produits) {
		this.produits = produits;
	}
	@Override
	public String toString() {
		return "Stock [id=" + id + ", code=" + code + ", adresse=" +
 adresse + "]";
	}
	public Stock() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Stock(String code, String adresse) {
		super();
		this.code = code;
		this.adresse = adresse;
	}
	public String getAdresse() {
		return adresse;
	}
	public void setAdresse(String adresse) {
		this.adresse = adresse;
	}
}
```

6. Lancer l'exécution du projet et remarquer :
   - ✓ La génération d'une table «**Responsable**» ayant les attributs de la classe « **Personne** » (id, nom et prenom) :

	#	Nom	Type
☐	1	id	bigint(20)
☐	2	nom	varchar(255)
☐	3	prenom	varchar(255)
☐	4	salaire	double

✓ La génération d'une table «**Employe**» ayant les attributs de la classe « **Personne** » (id, nom et prenom) et une clé étrangère (stock_id) :



	#	Nom	Type
☐	1	id	bigint(20)
☐	2	nom	varchar(255)
☐	3	prenom	varchar(255)
☐	4	nombre_heures	int(11)
☐	5	numero	int(11)
☐	6	stock_id	bigint(20)

7. Passer maintenant au package «**repository**» pour réaliser des traitements sur les données. Créer une interface «**PersonneRepository**» qui hérite de l'interface «**JpaRepository**». Cette interface offre les méthodes pour gérer toutes les entités dérivées de la classe «**Personne**». Voici sont code :

```
package spring.jpa.repository;

import
org.springframework.data.jpa.repository.JpaRepository;

import spring.jpa.model.Personne;

public interface PersonneRepository extends
JpaRepository<Personne, Long> {}
```

8. Reprendre le même exemple implémenté dans la classe «**JpaSpringBootApplication**» et ajouter les traitements suivants :
   a) Ajouter l'employé "**Triki**" "**Samir** ayant le numéro " **1**" au stock de "**Gabes**" :

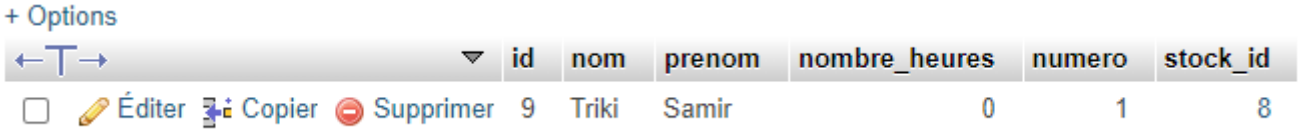
```
Employe e1 =  new Employe("Triki", "Samir", 1);
  // Récupérer une référence au stock de "Gabes"
  Stock sg = stockRepos.findByAdresse("Gabes");
  e1.setStock(sg);
  personneRepos.save(e1);
```

NB : Il est nécessaire de récupérer une implémentation de l'interface "**PersonneRepository**" nommée « **personneRepos** »

NB : il est nécessaire, aussi, de déclarer une méthode requête pour rechercher un stock par « **adresse** » dans l'interface « **StockRepository** » :

```
Stock findByAdresse(String adresse);
```

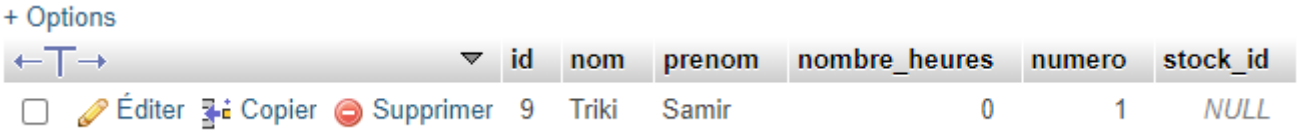
b) Lancer l'exécution et visualiser le contenu de la table «**employe**» :



c) Supprimer le stock de "**Gabes**" sans supprimer les employés associés :

```
if (sg!=null) // référence sur le stock de « Gabes »
{
     // récupérer tous les employé de Gabes
     Collection<Employe> le =sg.getEmployes();
     for( Employe e : le)
     {
          //Détacher l'employé du stock
          e.setStock(null);
          // enregistrer dans la table "Employe"
          personneRepos.saveAndFlush(e);
     }

     // vider la liste des employés
     le.clear();
     // supprimer le stock du "Gabes"
     stockRepos.delete(sg);
 }
```

d) Lancer l'exécution et visualiser le contenu de la table «**employe**» :



e) Affecter tous les employés libres (non associés à un stock) au stock de "Tunis"

```
//Récupérer le stock de "Tunis"
 Stock st = stockRepos.findByAdresse("Tunis");
 // Récupérer tous les employés non associés à un stock donné
 Collection<Employe> le =personneRepos.findAllByStockIsNull();
 if (le!=null)
```

```
        {
            for (Employe e: le)
            {
                //Affecter l'employé au stock de "Tunis"
                e.setStock(st);
                personneRepos.saveAndFlush(e);
            }
        }
```

NB : il est nécessaire, aussi, de déclarer, dans l'interface «**PersonneRepository**», une méthode requête pour retourner tous les employés dont le «**stock**» est **null** :

```
List <Employe> findAllByStockIsNull();
```

f) Lancer l'exécution et visualiser le contenu de la table «**employe**» :

+ Options

	id	nom	prenom	nombre_heures	numero	stock_id
Éditer Copier Supprimer	9	Triki	Samir	0	1	1