Relation N-M

Démonstration 8 du module 5

Les objectifs de cette démonstration sont

- Mise en place d'une relation M-N unidirectionnelle entre les personnes et leurs pays visités
- Mise en place d'une relation M-N bidirectionnelle les personnes et leurs pays visités

Contexte

- Continuer dans le projet précédent
- La liste des pays à visiter est : France, Espagne, Portugal, Italie et Grèce.
    - On est dans le cas d'un voyage organiser, où il est possible de choisir ses escales

Déroulement

1. Relation M-N unidirectionnelle

- Renommer le package des associations précédentes en com
- Dans la classe Personne
    - Ajout d'un nom spécifique sur l'entité et pour la table
    - Ajout d'attribut paysVisites qui représente la liste des pays qu'une personne a pu visiter et initialiser la liste dans les constructeurs.
    - Ajouter 2 méthodes pour gérer l'ajout et la suppression d'un pays
        - Dans une relation M-N, la suppression doit toujours être gérée de manière programmatique car plus complexe que dans les cas 1-N
    - Placer l'annotation @ManyToMany
        - Pas de gestion de cascade par l'ORM. Il y a trop de contraintes à partager la même entité sur des relations N-M il faut être très prudent. Il vaut mieux gérer cela de manière programmatique
        - FetchType.EAGER → pour charger tous les éléments de l'association

```
package fr.eni.demo.mtm.uni;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
```

```
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity(name = "personne_mtm_u")
@Table(name = "personne_mtm_u")
public class Personne {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private long id;

	private String nom;
	private String prenom;

	@ManyToMany(fetch = FetchType.EAGER) //pas cascade
	@JoinTable(name="PersonnePays",
			joinColumns= {@JoinColumn(name="personne_id")},
			inverseJoinColumns= {@JoinColumn(name="paysVisites_id")}
	)
	private List<Pays> paysVisites;

	public Personne() {
		paysVisites = new ArrayList<Pays>();
	}

	public Personne(String nom, String prenom) {
		this();
		this.nom = nom;
		this.prenom = prenom;
	}


	public void addPaysVisites(Pays a) {
		paysVisites.add(a);
	}


	public void removePaysVisites(Pays p) {
		getPaysVisites().remove(p);
	}

	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 prénom) {
		this.prenom = prénom;
	}
```

```
	public List<Pays> getPaysVisites() {
		return paysVisites;
	}

	public void setPaysVisites(List<Pays> paysVisites) {
		this.paysVisites = paysVisites;
	}

	@Override
	public String toString() {
		return "Personne [id=" + id + ", nom=" + nom + ", prenom=" + prenom + ", paysVisites=" +
paysVisites + "]";
	}

}
```

- Dans la classe Pays
  - Ajout d'un nom spécifique sur l'entité et pour la table
  - La clef primaire est un string qui sera gérée de manière programmatique

```
package fr.eni.demo.mtm.uni;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity(name = "pays_mtm_u")
@Table(name = "pays_mtm_u")
public class Pays {

	@Id
	private String cle;

	private String libelle;

	public Pays() {
	}

	public Pays(String cle, String libelle) {
		this.cle = cle;
		this.libelle = libelle;
	}

	public String getCle() {
		return cle;
	}

	public void setCle(String cle) {
		this.cle = cle;
	}

	public String getLibelle() {
		return libelle;
	}

	public void setLibelle(String libelle) {
		this.libelle = libelle;
	}

	@Override
	public String toString() {
```

```
		return "Pays [cle=" + cle + ", libelle=" + libelle + "]";
	}
}
```

- Création des 2 Repository

```
package fr.eni.demo.mtm.uni;

import org.springframework.data.repository.CrudRepository;

public interface PersonneMTMURepository extends CrudRepository<Personne, Long>{

}
```

```
package fr.eni.demo.mtm.uni;

import org.springframework.data.repository.CrudRepository;

public interface PaysMTMURepository extends CrudRepository<Pays, String>{

}
```

- Dans la classe d'exécution de l'application
  - Positionner l'annotation @Profile(«demo») sur le bean précédent et renommer le package com des entités
  - Copier le code du nouveau bean :

```
@Bean
public CommandLineRunner demoManyToMany(PersonneMTMURepository persDAO, PaysMTMURepository paysDAO ) {
	return (args) -> {

		List<fr.eni.demo.mtm.uni.Pays> paysAVisiter = new ArrayList<>();
			paysAVisiter.add(new fr.eni.demo.mtm.uni.Pays("fr", "France"));
			paysAVisiter.add(new fr.eni.demo.mtm.uni.Pays("it", "Italie"));
			paysAVisiter.add( new fr.eni.demo.mtm.uni.Pays("gr", "Grece"));
			paysAVisiter.add( new fr.eni.demo.mtm.uni.Pays("es", "Espagne"));
			paysAVisiter.add( new fr.eni.demo.mtm.uni.Pays("po", "Portugal"));

			paysDAO.saveAll(paysAVisiter);


	fr.eni.demo.mtm.uni.Personne albert = new fr.eni.demo.mtm.uni.Personne("Dupontel", "Albert");
	fr.eni.demo.mtm.uni.Personne sophie = new fr.eni.demo.mtm.uni.Personne("Marceau", "Sophie");

			albert.addPaysVisites(paysAVisiter.get(0));
			albert.addPaysVisites(paysAVisiter.get(1));
			albert.addPaysVisites(paysAVisiter.get(2));
			albert.addPaysVisites(paysAVisiter.get(3));

			sophie.addPaysVisites(paysAVisiter.get(2));
			sophie.addPaysVisites(paysAVisiter.get(4));

			persDAO.save(albert);
			persDAO.save(sophie);

			System.out.println("Liste des personnes : ");
			System.out.println("-------------------------------");
			for (fr.eni.demo.mtm.uni.Personne personne : persDAO.findAll()) {
				System.out.println(personne.toString());
			}

			System.out.println("Suppression d'un pays pour albert");
			System.out.println("-------------------------------");
```

```
                albert.removePaysVisites(paysAVisiter.get(2));
                persDAO.save(albert);
        Optional<fr.eni.demo.mtm.uni.Personne> opt = persDAO.findById(Long.valueOf(albert.getId()));
                if(opt.isPresent()) {
                        System.out.println(opt.get());
                }
            };
        }
```

- Traces d'exécution :

```
…
Hibernate:

    create table pays_mtm_u (
       cle varchar(255) not null,
        libelle varchar(255),
        primary key (cle)
    ) engine=InnoDB
Hibernate:

    create table personne_mtm_u (
       id bigint not null auto_increment,
        nom varchar(255),
        prenom varchar(255),
        primary key (id)
    ) engine=InnoDB
Hibernate:

    create table personne_pays (
       personne_id bigint not null,
        pays_visites_id varchar(255) not null
    ) engine=InnoDB
Hibernate:

    alter table personne_pays
       add constraint FK2w448ye5ckc0nujmd17310255
       foreign key (pays_visites_id)
       references pays_mtm_u (cle)
Hibernate:

    alter table personne_pays
       add constraint FK1kwchoaitpq5mqb20pb4ncx3n
       foreign key (personne_id)
       references personne_mtm_u (id)
…
Hibernate:
    insert
    into
        personne_mtm_u
        (nom, prenom)
    values
        (?, ?)
Hibernate:
    insert
    into
        pays_mtm_u
        (libelle, cle)
    values
        (?, ?)
Hibernate:
    insert
    into
        pays_mtm_u
        (libelle, cle)
    values
```

```
        (?, ?)
Hibernate:
    insert
    into
        pays_mtm_u
        (libelle, cle)
    values
        (?, ?)
Hibernate:
    insert
    into
        pays_mtm_u
        (libelle, cle)
    values
        (?, ?)
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
…
Liste des personnes :
------------------------------
Hibernate:
    select
        personne0_.id as id1_1_,
        personne0_.nom as nom2_1_,
        personne0_.prenom as prenom3_1_
    from
        personne_mtm_u personne0_
Hibernate:
    select
        paysvisite0_.personne_id as personne1_2_0_,
        paysvisite0_.pays_visites_id as pays_vis2_2_0_,
        pays1_.cle as cle1_0_1_,
        pays1_.libelle as libelle2_0_1_
    from
        personne_pays paysvisite0_
    inner join
        pays_mtm_u pays1_
            on paysvisite0_.pays_visites_id=pays1_.cle
    where
        paysvisite0_.personne_id=?
```

```
Hibernate:
    select
        paysvisite0_.personne_id as personne1_2_0_,
        paysvisite0_.pays_visites_id as pays_vis2_2_0_,
        pays1_.cle as cle1_0_1_,
        pays1_.libelle as libelle2_0_1_
    from
        personne_pays paysvisite0_
    inner join
        pays_mtm_u pays1_
            on paysvisite0_.pays_visites_id=pays1_.cle
    where
       paysvisite0_.personne_id=?
Personne [id=1, nom=Dupontel, prenom=Albert, paysVisites=[Pays [cle=fr, libelle=France], Pays [cle=it,
libelle=Italie], Pays [cle=gr, libelle=Grece], Pays [cle=es, libelle=Espagne]]]
Personne [id=2, nom=Marceau, prenom=Sophie, paysVisites=[Pays [cle=gr, libelle=Grece], Pays [cle=po,
libelle=Portugal]]]
Suppression d'un pays pour albert
------------------------------
Hibernate:
    select
        personne0_.id as id1_1_1_,
        personne0_.nom as nom2_1_1_,
        personne0_.prenom as prenom3_1_1_,
        paysvisite1_.personne_id as personne1_2_3_,
        pays2_.cle as pays_vis2_2_3_,
        pays2_.cle as cle1_0_0_,
        pays2_.libelle as libelle2_0_0_
    from
        personne_mtm_u personne0_
    left outer join
        personne_pays paysvisite1_
            on personne0_.id=paysvisite1_.personne_id
    left outer join
        pays_mtm_u pays2_
            on paysvisite1_.pays_visites_id=pays2_.cle
    where
        personne0_.id=?
Hibernate:
    delete
    from
        personne_pays
    where
        personne_id=?
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
Hibernate:
    insert
    into
        personne_pays
        (personne_id, pays_visites_id)
    values
        (?, ?)
Hibernate:
```

```
    select
        personne0_.id as id1_1_0_,
        personne0_.nom as nom2_1_0_,
        personne0_.prenom as prenom3_1_0_,
        paysvisite1_.personne_id as personne1_2_1_,
        pays2_.cle as pays_vis2_2_1_,
        pays2_.cle as cle1_0_2_,
        pays2_.libelle as libelle2_0_2_
    from
        personne_mtm_u personne0_
    left outer join
        personne_pays paysvisite1_
            on personne0_.id=paysvisite1_.personne_id
    left outer join
        pays_mtm_u pays2_
            on paysvisite1_.pays_visites_id=pays2_.cle
    where
       personne0_.id=?
Personne [id=1, nom=Dupontel, prenom=Albert, paysVisites=[Pays [cle=fr, libelle=France], Pays [cle=it,
libelle=Italie], Pays [cle=es, libelle=Espagne]]]
```

- Création 3 tables avec une table de jointure
    - Et 2 contraintes ; une pour chaque table
    - En base, on retrouve bien cela

- Les enregistrements en base :

Table des pays

cle	libelle
es	Espagne
fr	France
gr	Grece
it	Italie
po	Portugal

Table des personnes

id	nom	prenom
1	Dupontel	Albert
2	Marceau	Sophie

Table de jointure

personne_id	pays_visites_id
2	gr
2	po
1	fr
1	it
1	es

- Pour Hibernate, pour la sauvegarde d'une personne et de ses pays visités.
    - Il doit mettre un insert sur les tables pays et personne, et puis un sur la table de jointure
- Pour la suppression d'un pays sur une personne
    - Hibernate supprime toutes les clefs de la table de jointure correspondantes à cette personne
    - Puis recrée celle restante.

2. Relation M-N bidirectionnelle

- Dans ce cas, il y a dans chaque classe, un attribut de l'autre classe
- Pour éviter que l'ORM ne passe son temps à aller d'association en association (boucler)
    - Il faut préciser que l'ORM ne regarde qu'un côté de l'association.
    - Pour cela, il faut utiliser l'attribut mappedBy.
        - Choisir celui, le moins appelé au niveau métier.
    - Il faut aussi ajouter le comportement pour gérer la bidirectionnalité en Java (comme

pour le cas 1-1)

- Dupliquez les classes précédentes
- Dans la classe Pays :
  - Changer le nom de la table et de l'entité
  - Ajouter le nouvel attribut et son annotation avec mappedBy (Getter/Setter)
    - Ajouter fetch en EAGER pour remonter dans les 2 sens les informations depuis la base
  - Ajouter l'initialisation de la liste dans les constructeurs
  - Ajouter une méthode pour gérer l'ajout d'une personne pour la bidirectionnalité

```
package fr.eni.demo.mtm.bi;

import java.util.List;

import javax.persistence.*;

@Entity(name = "pays_mtm_bi")
@Table(name = "pays_mtm_bi")
public class Pays {

	@Id
	private String cle;

	private String libelle;

	@ManyToMany(mappedBy="paysVisites", fetch = FetchType.EAGER)
	private List<Personne> personnes;

	public Pays() {
		personnes = new ArrayList<Personne>();
	}

	public Pays(String cle, String libelle) {
		this();
		this.cle = cle;
		this.libelle = libelle;
	}

	public List<Personne> getPersonnes() {
		return personnes;
	}

	//Gestion de la bidirectionnalité
	public void addPersonne(Personne personne) {
		if(!this.personnes.contains(personne)){
			this.personnes.add(personne);
		}
	}

	public String getCle() {
		return cle;
	}

	public void setCle(String cle) {
		this.cle = cle;
	}

	public String getLibelle() {
		return libelle;
	}
```

```
	public void setLibelle(String libelle) {
		this.libelle = libelle;
	}

	@Override
	public String toString() {
		return "Pays [cle=" + cle + ", libelle=" + libelle + "]";
	}
}
```

- Dans la classe Personne
  - Changer le nom de la table et de l'entité

    Et ajouter le code pour gérer la bidirectionnalité dans les méthode addPaysVisites et removePaysVisistes

```
…
@Entity(name = "personne_mtm_bi")
@Table(name = "personne_mtm_bi")
public class Personne {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private long id;

	private String nom;
	private String prenom;

	@ManyToMany(fetch = FetchType.EAGER) //pas cascade
	@JoinTable(name="PersonnePays",
			joinColumns= {@JoinColumn(name="personne_id")},
			inverseJoinColumns= {@JoinColumn(name="paysVisites_id")}
	)
	private List<Pays> paysVisites;

	…

	public void addPaysVisites(Pays a) {
		paysVisites.add(a);
		//Gestion de la bidirectionnalité
		a.addPersonne(this);
	}

	public void removePaysVisites(Pays p) {
		getPaysVisites().remove(p);
		//Gestion de la bidirectionnalité
		p.getPersonnes().remove(this);
	}
…
```

- Création des 2 Repository

```
package fr.eni.demo.otm.bi;

import org.springframework.data.repository.CrudRepository;

public interface PersonneMTMBiRepository extends CrudRepository<Personne, Long>{

}
```

```
package fr.eni.demo.otm.bi;

import org.springframework.data.repository.CrudRepository;

public interface PaysMTMBiRepository extends CrudRepository<Pays, String>{

}
```

- Dans la classe d'exécution de l'application
  - Positionner l'annotation @Profile(«demo») sur le bean précédent et renommer le package com des entités
  - Copier le code du nouveau bean :

```
@Bean
public CommandLineRunner demoManyToManyBi(PersonneMTMBiRepository persDAO, PaysMTMBiRepository paysDAO )
{
	return (args) -> {
		fr.eni.demo.mtm.bi.Pays fr = new fr.eni.demo.mtm.bi.Pays("fr", "France");
		fr.eni.demo.mtm.bi.Pays it = new fr.eni.demo.mtm.bi.Pays("it", "Italie");
		fr.eni.demo.mtm.bi.Pays gb = new fr.eni.demo.mtm.bi.Pays("gb", "Grande Bretagne");
		fr.eni.demo.mtm.bi.Pays gr = new fr.eni.demo.mtm.bi.Pays("gr", "Grece");
		fr.eni.demo.mtm.bi.Pays es = new fr.eni.demo.mtm.bi.Pays("es", "Espagne");
		fr.eni.demo.mtm.bi.Pays po = new fr.eni.demo.mtm.bi.Pays("po", "Portugal");

		fr.eni.demo.mtm.bi.Personne albert = new fr.eni.demo.mtm.bi.Personne("Dupontel", "Albert");
		fr.eni.demo.mtm.bi.Personne sophie = new fr.eni.demo.mtm.bi.Personne("Marceau", "Sophie");

			albert.addPaysVisites(fr);
			albert.addPaysVisites(gb);
			albert.addPaysVisites(po);
			albert.addPaysVisites(es);

			sophie.addPaysVisites(it);
			sophie.addPaysVisites(gr);

			persDAO.save(albert);
			persDAO.save(sophie);

			System.out.println("Liste des personnes : ");
			System.out.println("-------------------------------");
			for (fr.eni.demo.mtm.bi.Personne personne : persDAO.findAll()) {
				System.out.println(personne.toString());
			}

			System.out.println("Suppression d'un pays pour albert");
			System.out.println("-------------------------------");
			albert.removePaysVisites(po);
			persDAO.save(albert);
		Optional<fr.eni.demo.mtm.bi.Personne> opt = persDAO.findById(Long.valueOf(albert.getId()));
			if(opt.isPresent()) {
				System.out.println(opt.get());
			}

			System.out.println("Récupération des personnes se rendant en Grèce");
			System.out.println("-------------------------------");
		Optional<fr.eni.demo.mtm.bi.Pays> opt2 = paysDAO.findById(paysAVisiter.get(2).getCle());
			if(opt2.isPresent()) {
				fr.eni.demo.mtm.bi.Pays p = opt2.get();
				System.out.println(p.getPersonnes());
			}
	};
}
```

- Les traces et les tables générées sont pratiquement identiques avec la version unidirectionnelle.
- Les différences sont uniquement sur le nommage des tables et des entités.
- Et sur la dernière partie, où nous cherchons à remonter les personnes qui se rendent en Grèce :

```
Récupération des personnes se rendant en Grèce
------------------------------
Hibernate:
    select
        pays0_.cle as cle1_0_0_,
        pays0_.libelle as libelle2_0_0_,
        personnes1_.pays_visites_id as pays_vis2_2_1_,
        personne2_.id as personne1_2_1_,
        personne2_.id as id1_1_2_,
        personne2_.nom as nom2_1_2_,
        personne2_.prenom as prenom3_1_2_
    from
        pays_mtm_bi pays0_
    left outer join
        personne_pays personnes1_
            on pays0_.cle=personnes1_.pays_visites_id
    left outer join
        personne_mtm_bi personne2_
            on personnes1_.personne_id=personne2_.id
    where
        pays0_.cle=?
Hibernate:
    select
        paysvisite0_.personne_id as personne1_2_0_,
        paysvisite0_.pays_visites_id as pays_vis2_2_0_,
        pays1_.cle as cle1_0_1_,
        pays1_.libelle as libelle2_0_1_
    from
        personne_pays paysvisite0_
    inner join
        pays_mtm_bi pays1_
            on paysvisite0_.pays_visites_id=pays1_.cle
    where
        paysvisite0_.personne_id=?
Hibernate:
    select
        personnes0_.pays_visites_id as pays_vis2_2_0_,
        personnes0_.personne_id as personne1_2_0_,
        personne1_.id as id1_1_1_,
        personne1_.nom as nom2_1_1_,
        personne1_.prenom as prenom3_1_1_
    from
        personne_pays personnes0_
    inner join
        personne_mtm_bi personne1_
            on personnes0_.personne_id=personne1_.id
    where
        personnes0_.pays_visites_id=?
[Personne [id=2, nom=Marceau, prenom=Sophie, paysVisites=[Pays [cle=gr, libelle=Grece], Pays [cle=po,
libelle=Portugal]]]]
```

  - Il ne reste plus que sophie, car albert ne désire plus aller en Grèce.
  - Toutes les fonctionnalités et la bidirectionnalité sont opérationnelles.