Clefs primaires composites

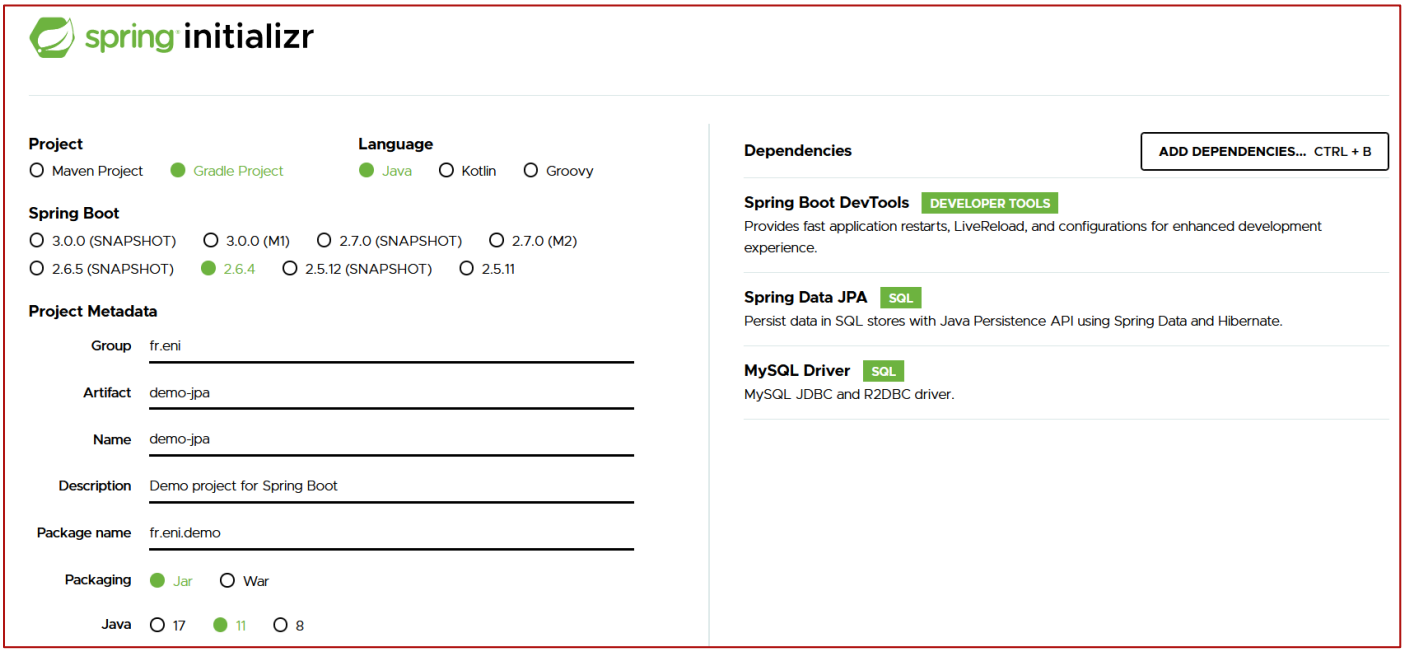
Démonstration 4 du module 5

Les objectifs de cette démonstration sont

- Créer une clef primaire composite à l'aide des annotations @Id et @IdClass
- Créer une clef primaire composite à l'aide des annotations @EmbeddedId et @Embeddable

Contexte

- Création d'un nouveau projet
- Avec les starters :
  - Spring Boot Data JPA
  - Spring Boot Dev Tools
- Et le driver à la base de données MySQL Server



- Importer le projet dans IDE
- Recopier la configuration de la base de données dans application.properties.

Déroulement

1. Clefs primaires composites : @Id et @IdClass

Le but est de créer une classe Personne qui a pour clef primaire les attributs nom et prenom.

- Créer la classe qui représente la clef composite :

```
package fr.eni.demo.key1;

import java.io.Serializable;

public class PersonnePK implements Serializable {
	private static final long serialVersionUID = 1L;
	private String nom;
	private String prenom;

	public PersonnePK() {
	}

	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;
	}
}
```

- Ajout de la classe principale,
  - avec @Id positionnée sur les 2 attributs correspondant.
  - Et l'annotation @IdClass pour liée les 2 attributs à la classe

```
package fr.eni.demo.key1;

import java.time.LocalDate;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.IdClass;
import javax.persistence.Table;
import javax.persistence.Transient;

@Entity(name = "personne_pk_1")
@IdClass(PersonnePK.class)
@Table(name = "personne_pk_1")
public class Personne {

	@Id
	private String nom;
	@Id
	private String prenom;

	private LocalDate dateNaissance;

	@Transient
```

```
	private int age;

	public Personne() {
	}

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

	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 LocalDate getDateNaissance() {
		return dateNaissance;
	}

	public void setDateNaissance(LocalDate dateNaissance) {
		this.dateNaissance = dateNaissance;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

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

- Création d'un Repository pour gérer l'entité

```
package fr.eni.demo.key1;

import org.springframework.data.repository.CrudRepository;

public interface PersonnePK1Repository extends CrudRepository<Personne, PersonnePK>{

}
```

- Dans la classe d'exécution de l'application, ajout d'un bean pour tester la configuration :

```
@Bean
public CommandLineRunner demoKey1(PersonnePK1Repository repository) {
	return (args) -> {
// save a few customers
repository.save(new fr.eni.demo.key1.Personne("Legrand", "Lucie", LocalDate.parse("2008-06-18")));
repository.save(new fr.eni.demo.key1.Personne("Legrand", "Lucie2", LocalDate.parse("2006-04-03")));

		// fetch all customers
		// sysout //("Customers found with findAll():");
		System.out.println("Liste des personnes : ");
		System.out.println("------------------------------");
		for (fr.eni.demo.key1.Personne p : repository.findAll()) {
			System.out.println(p.toString());
		}
	};
}
```

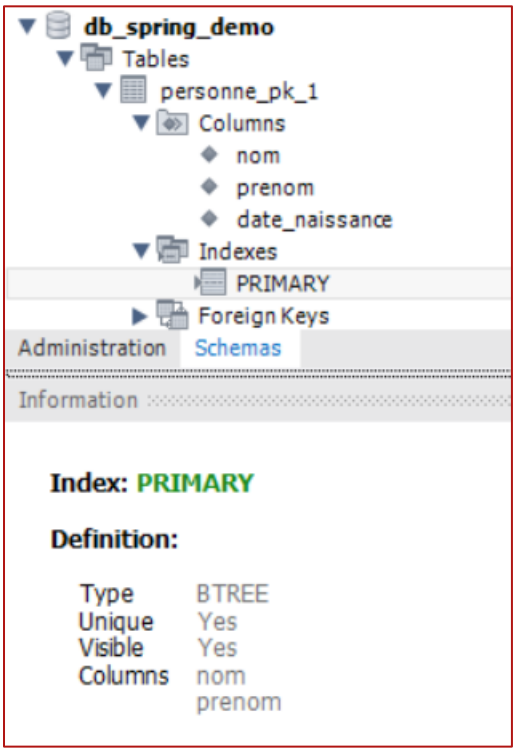
- Traces d'exécution :

```
Hibernate:

    drop table if exists personne_pk_1
Hibernate:

    create table personne_pk_1 (
       nom varchar(255) not null,
        prenom varchar(255) not null,
        date_naissance date,
        primary key (nom, prenom)
    ) engine=InnoDB
…
Hibernate:
    select
        personne0_.nom as nom1_0_0_,
        personne0_.prenom as prenom2_0_0_,
        personne0_.date_naissance as date_nai3_0_0_
    from
        personne_pk_1 personne0_
    where
        personne0_.nom=?
        and personne0_.prenom=?
Hibernate:
    insert
    into
        personne_pk_1
        (date_naissance, nom, prenom)
    values
        (?, ?, ?)
…
Liste des personnes :
------------------------------
Hibernate:
    select
        personne0_.nom as nom1_0_,
        personne0_.prenom as prenom2_0_,
        personne0_.date_naissance as date_nai3_0_
    from
        personne_pk_1 personne0_
Personne [nom=Legrand, prenom=Lucie, dateNaissance=2008-06-18, age=0]
Personne [nom=Legrand, prenom=Lucie2, dateNaissance=2006-04-03, age=0]
```

- En base; création d'une table avec la clef primaire composée des 2 colonnes :



- Dans cette version, tous les attributs restent dans la classe d'entité principale : People.

2. Clefs primaires composites : @EmbeddedId et @Embeddable

- Créer la classe qui représente la clef composite est identique à la précédente, il suffit de poser l'annotation : @Embeddable sur la classe
- Par contre, la classe Personne n'a plus ses attributs nom et prenom mais un personnePK annoté @EmbeddedId

```
package fr.eni.demo.key2;

import java.time.LocalDate;

import javax.persistence.EmbeddedId;
import javax.persistence.Entity;
import javax.persistence.Table;
import javax.persistence.Transient;

@Entity(name = "personne_pk_2")
@Table(name = "personne_pk_2")
public class Personne {

	@EmbeddedId
	private PersonnePK personnePK;

	private LocalDate dateNaissance;

	@Transient
	private int age;

	public Personne() {
	}

	public Personne(String nom, String prenom, LocalDate dateNaissance) {
		personnePK = new PersonnePK();
		personnePK.setNom(prenom);
		personnePK.setPrenom(nom);
		this.dateNaissance = dateNaissance;
	}

	public String getPrenom() {
		return personnePK.getPrenom();
	}

	public String getNom() {
		return personnePK.getNom();
	}

	public PersonnePK getPersonnePK() {
		return personnePK;
	}

	public void setPersonnePK(PersonnePK peoplePK) {
		this.personnePK = peoplePK;
	}

	public LocalDate getDateNaissance() {
		return dateNaissance;
	}

	public void setDateNaissance(LocalDate dateNaissance) {
		this.dateNaissance = dateNaissance;
	}

	public int getAge() {
		return age;
	}
```

```
	public void setAge(int age) {
		this.age = age;
	}

	@Override
	public String toString() {
		return "Personne [nom=" + getNom() + ", prenom=" + getPrenom() + ", dateNaissance=" +
dateNaissance + ", age="
				+ age + "]";
	}

}
```

- Dans la classe d'exécution de l'application
  - Pour éviter de créer plusieurs projets nous allons exploiter 2 astuces de Spring
    - La première que nous avons déjà vu est de placer `@Profile("demo")` sur les beans qui ne seront plus utilisés.
    - Pour les entités, renommer le package par défaut en com. La norme de Spring est de regarder les sous packages de la classe Main. Donc elle n'ira pas regarder les com…
  - Modifier le code comme suit :

```
@Bean
public CommandLineRunner demoKey2(PersonnePK2Repository repository) {
	return (args) -> {
	// save a few customers
	repository.save(new fr.eni.demo.key2.Personne("Legrand", "Lucie", LocalDate.parse("2008-06-18")));
	repository.save(new fr.eni.demo.key2. Personne("Legrand", "Lucie2", LocalDate.parse("2006-04-03")));

	// fetch all customers
	// sysout //("Customers found with findAll():");
	System.out.println("Liste des personnes : ");
	System.out.println("-------------------------------");
		for (fr.eni.demo.key2.Personne p : repository.findAll()) {
			System.out.println(p.toString());
		}
	};
}
```

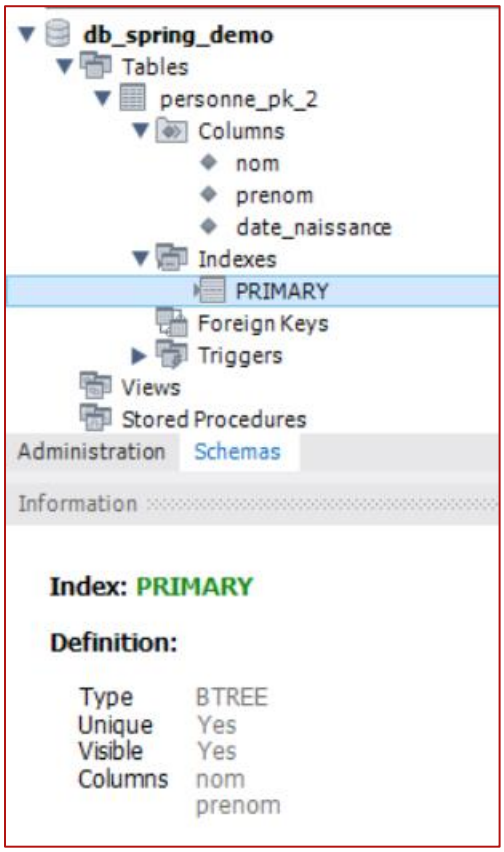
- Traces d'exécution, sont similaires aux précédentes :

```
Hibernate: drop table if exists people_pk_1 //ATTENTION, il gère à chaque fois les annotations @Entity
Hibernate: drop table if exists personne_pk_2
…
Hibernate:

    create table personne_pk_2 (
       nom varchar(255) not null,
        prenom varchar(255) not null,
        date_naissance date,
        primary key (nom, prenom)
    ) engine=InnoDB
…
Hibernate:
    select
        personne0_.nom as nom1_1_0_,
        personne0_.prenom as prenom2_1_0_,
        personne0_.date_naissance as date_nai3_1_0_
    from
        personne_pk_2 personne0_
    where
```

```
        personne0_.nom=?
        and personne0_.prenom=?
Hibernate:
    insert
    into
        personne_pk_2
        (date_naissance, nom, prenom)
    values
        (?, ?, ?)
…
Liste des personnes :
------------------------------
Hibernate:
    select
        personne0_.nom as nom1_1_,
        personne0_.prenom as prenom2_1_,
        personne0_.date_naissance as date_nai3_1_
    from
        personne_pk_2 personne0_
Personne [nom=Lucie, prenom=Legrand, dateNaissance=2008-06-18, age=0]
Personne [nom=Lucie2, prenom=Legrand, dateNaissance=2006-04-03, age=0]
```

- En base; création d'une table avec la clef primaire composée des 2 colonnes, la structure est la même que précédemment :



La seule contrainte de cette solution par rapport à l'autre est le changement de structure sur l'entité principale