**⛔ Erreur**



**💡Solution**

Il faut ajouter l’annotation @ToString.Exclude pour les associations

@Entity

@Getter

@Setter

**@ToString**

@NoArgsConstructor

@AllArgsConstructor

**public class** Apprenant **implements** Serializable {

@Id

@GeneratedValue(strategy= GenerationType.***IDENTITY***)

**private** Integer **idApprenant**;

**@JsonIgnore**

**@ToString.Exclude**

@ManyToMany(mappedBy = **"listApprenants"**)

**private** Set<Formation> **listFormations**;

**⛔ Erreur**

Failed to convert from type [java.lang.String] to type [@org.springframework.web.bind.annotation.PathVariable java.time.LocalDate]

**💡Solution**

* Controller: @PathVariable( "startDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate startDate

@DateTimeFormat(iso = DateTimeFormat.ISO.DATE)

@GetMapping(**"GetSubscriptionByStartDate/{startDate}"**)

**public** List<Abonnement> retrieveSubscriptionsByDate(@PathVariable(**"startDate"**) **@DateTimeFormat(iso = DateTimeFormat.ISO.*DATE*)** LocalDate startDate) {

**return iAbonnementService**.retrieveSubscriptionsByDates(startDate);

}}

Ou bien

application.properties

**spring.mvc.format.date**= **yyyy-MM-dd**

**⛔ Erreur**

Error failed to lazily initialize a collection



OneToMany et ManyToMany : LAZY

**💡Solution**

@ManyToMany(fetch=FetchType.***EAGER***)

**⛔ Erreur**

Null affectation list



public MatchFootball ajouterMatchFootballEtAffecterEquipe

(MatchFootball matchFootball, Integer idEquipe1, Integer idEquipe2)

**💡Solution**

**@ManyToMany**(fetch = FetchType.***EAGER***)

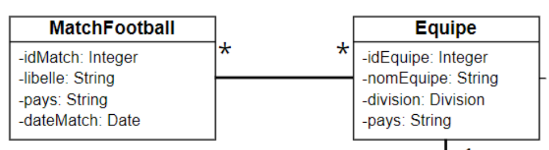
@JsonIgnore

**private** Set<Equipe> **equipes** = **new** HashSet<>();

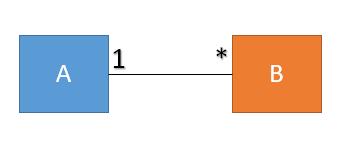
ou bien

**private** List<Equipe> **equipes** = **new** ArrayList<>();

Pour affecter une équipe, on peut initialiser la liste des équipes au niveau match



**✏️Affectation**

****

Classe fille Classe mère => **responsable de l’affectation**

**InterfaceB** {

void assignAToB (Long idA, Long idB);

}

**Service**

**@Service**

**@AllArgsConstructor**

public class ServiceB implements **InterfaceB** {

private ARepository **aRepository**;

private BRepository **bRepository**;

void assignAToB (Long idA, Long idB) {

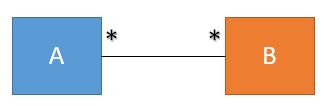
A a = **aRepository.findById(idA).orElse(null);**

**B b = bRepository.findById(idB).orElse(null);**

b.setA(a);

**bRepository.save(b);**

} }

****

Classe fille Classe mère => **responsable de l’affectation**

**InterfaceB** {

void assignAToB (Long idA, Long idB);

}

**Service**

**@Service**

**@AllArgsConstructor**

public class ServiceB implements **InterfaceB** {

private ARepository **aRepository**;

private BRepository **bRepository**;

void assignAToB (Long idA, Long idB){

A a = **aRepository.findById(idA).orElse(null);**

**B b = bRepository.findById(idB).orElse(null);**

b.getA().add(a);

**bRepository.save(b);**

}

}

**Autres remarques**

* Les énumérations doivent être stockées en tant que chaînes de caractères dans la base de données. => @Enumerated(EnumType.***STRING***)

Exemple:

@Enumerated(EnumType.***STRING***)

**private** Specialite **specialite**;

* **java.util.Date** que nous utilisons appartient à des versions inférieures à java 8.

Nous devons plutôt utiliser **java.time.LocalDate** car java8 nous offre des fonctions prédéfinies pour la manipulation des dates qui ne fonctionnent qu'avec ce type.



- Pour récupérer la date courante: **LocalDate.now()**

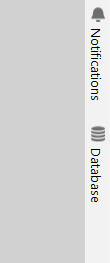
- Pour la différence en jours entre les dates: **ChronoUnit.DAYS.between(date1,date2)**

**- before** & **after** seront **isBefore** & **isAfter**

- L'annotation @Temporal ne sera plus nécessaire.

* Entity: @DateTimeFormat(pattern = "yyyy-MM-dd")

* Si vous ne trouvez pas maven lors de la création d'un projet

 => **Solution** :  puis : +Add as Maven Project

* Pour Spring scheduler il suffit juste d’ajouter l'annotation et l'expression au niveau du service. (sans controller)

* @Column(nullable = **false**, unique = **true**)

**private** String **code**;

attribut unique

attribut ne doit pas être null;

* @PostMapping(**"/add-Client"**)

**public void** ajouterEtAffecterClientBoutiques(

@RequestBody Client client, **@RequestParam**(value=**"idBoutiques"**) **List<Long>** idBoutiques) {

**clientService**.ajouterEtAffecterClientBoutiques(client, idBoutiques);

}

* **spring.mvc.format.date**= **yyyy-MM-dd**