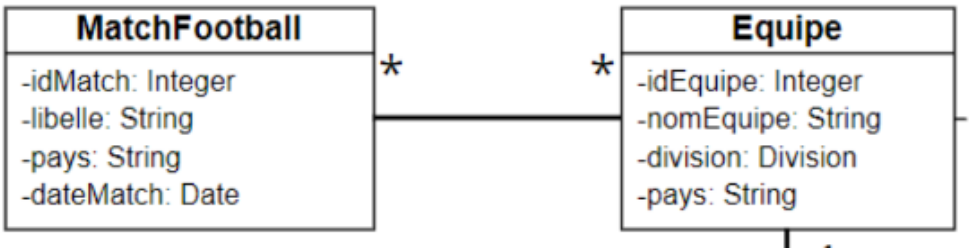
- Null affectation list
  public MatchFootball ajouterMatchFootballEtAffecterEquipe
  (MatchFootball matchFootball, Integer idEquipe1, Integer idEquipe2)

  Solution =>

```
@ManyToMany(fetch = FetchType.EAGER)
@JsonIgnore
private Set<Equipe> equipes = new HashSet<>();
```

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



- 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;
```

- OneToMany et ManyToMany : LAZY

```
failed to lazily initialize a collection of role: tn.esprit.spring.entities.Formation.list
```

  Error failed to lazily initialize a collection => **Solution**
  @ManyToMany(fetch=FetchType.*EAGER*)

- @DateTimeFormat(pattern = "yyyy-MM-dd")

- ```
  @Column(nullable = false, unique = true)
  private String code;
  ```

  attribut unique

  attribut ne doit pas être null;

- **Erreur:**
  ```

```
at java.lang.String.valueOf(String.java:2994)
```

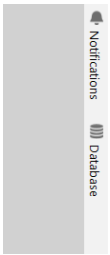
Solution:

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;
  private String nom;


  @JsonIgnore
  @ToString.Exclude
  @ManyToMany(mappedBy = "listApprenants")
  private Set<Formation> listFormations;
```

- 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)
- **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.

```
private LocalDate dateDebut;
private LocalDate dateFin;
```

- 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.

- @PathVariable(name = "endDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate endDate

  @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)