TP – Services Avancés : (Affectations / Désaffectations)

PLAN DU COURS

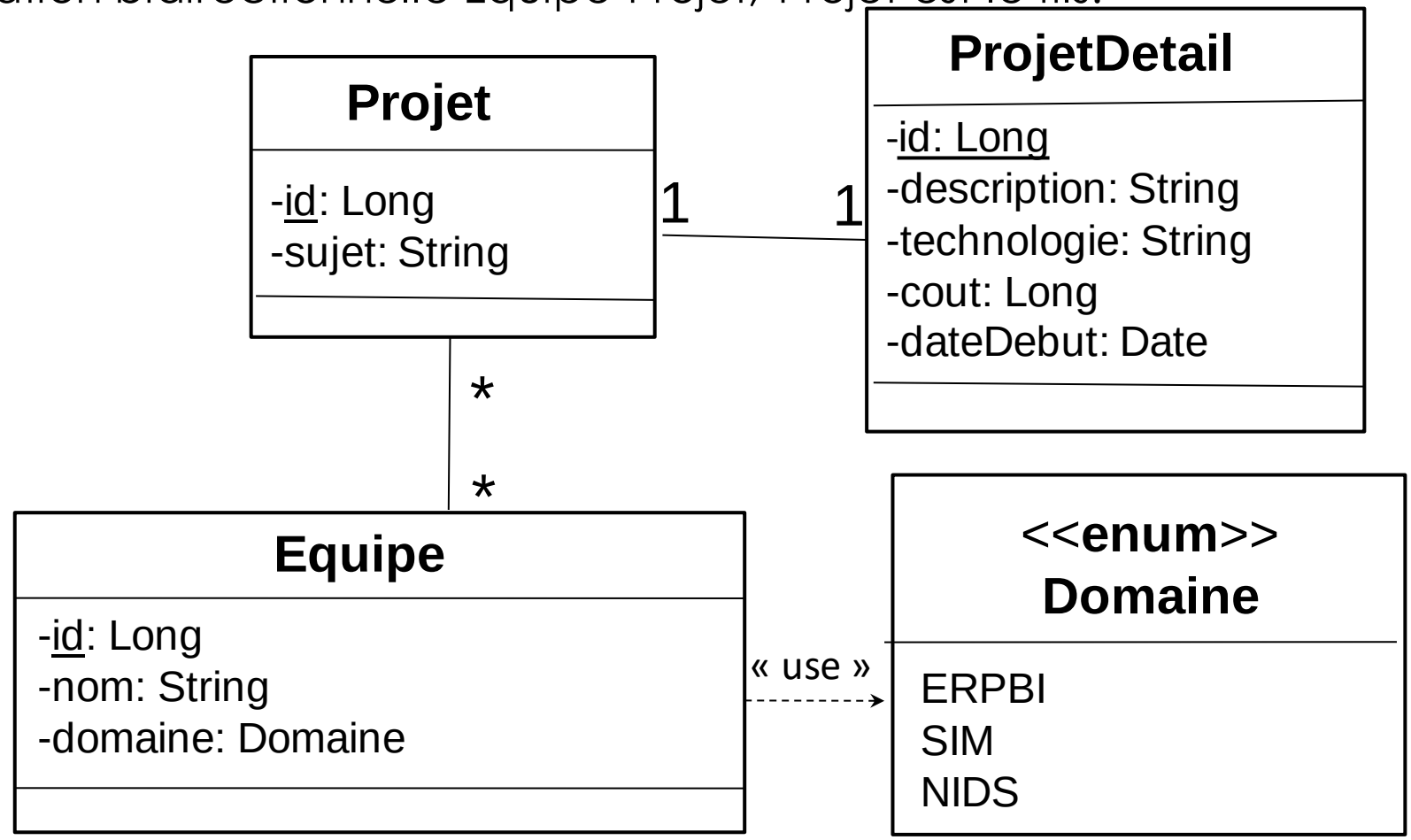
- **Cas 1 : Affectations (projet gestion-projet)**
- **Cas 2 : Affectations (projet gestion-projet)**
- **Cas 3 : Affectations (projet gestion-projet)**
- **Cas 4 : Affectations (projet gestion-projet)**

- **Cas 5 : Désaffectation (projet gestion-projet)**
- **Cas 6 : Désaffectation (projet gestion-projet)**

- **TP8 : Affectations et Désaffectations** liées au projet tp-foyer (Développement + Tests)

Affectations

Soit le diagramme de classes suivant (la séance dernière, nous avons déjà développés Entités, Associations et WS CRUD) :
Ce diagramme sera un exemple pour comprendre le mécanisme d'**Affectations**.
Dans la relation bidirectionnelle Projet-ProjetDetail, ProjetDetail est le fils.
Dans la relation bidirectionnelle Equipe-Projet, Projet est le fils.



Cas 1 : Affectation

On souhaite créer le service qui permet d'Ajouter un Projet et un Projet Detail et d'affecter le ProjetDetail au Projet

Solution :
- Dans l'entité Projet (Parent) ont doit cascader les actions du Projet vers le projetDetail
- Dans la méthode, on doit faire un ajout simple :

Cas 1 : Affectation

Service ProjetServiceImpl :

```
public Projet addProjetAndProjetDetailAndAssign(Projet  projet) {
    return projetRepository.save(projet);
}
```

Controller ProjetRestController :

http://localhost:8089/tp8/projet/ajouter-projet-et-projet-detail

```
@PostMapping("/ajouter-projet-et-projet-detail")
public Projet addProjetAndProjetDetail(@RequestBody Projet p) {
    Projet projet = projetService.addProjetAndProjetDetailAndAssign(p);
    return projet;
}
```

Cas 1 : Affectation

Entité Projet (**Parent**) : (l'ajout d'un projet provoque l'ajout d'un projet détail grâce à cascade)

```
@OneToOne(cascade = CascadeType.ALL)
  private ProjetDetail projetDetail;
```

Entité ProjetDetail (**fils**) :

```
@OneToOne(mappedBy = "projetDetail")
  @ToString.Exclude
  @JsonIgnore
  private Projet projet;
```

Cas 1 : Affectation

Objet JSON à utiliser : (à l'intérieur de l'objet projet, on met l'objet projet détail) :

```
{
  "sujet": "projet numéro 1",
  "projetDetail": {
    "description": "détail du projet",
    "technologie": "Java",
    "cout_provisoire": 1000,
    "dateDebut": "2022-11-14T23:52:48.343Z"
  }
}
```

Attention si Relation OneToMany ou ManyTomany :

```
{
" attribut1" : …,
" attribut2 " : [
{…} ,
{….}
]
}
```

Cas 1 : Affectation

Résultat dans la base de données :

Table t_projet_detail :

pd_id	pd_cout_provisoire	date_debut	pd_description	pd_technologie
16	0	2022-11-15	string	string

Table t_projet :

projet_id	projet_sujet	projet_detail_pd_id
2	string	16

Cas 2 : Affectation

On souhaite créer le service qui permet d'Affecter un ProjetDetail à un Projet (le ProjetDetail et le Projet sont déjà créés dans la base de données :

Solution :
Dans la méthode, on doit récupérer de la base de données le Projet et le ProjetDétail, puis affecter le ProjetDetail (fils) au Projet (parent), puis faire une mise à jour :

Cas 2 : Affectation

Service :

```
public void assignProjetDetailToProjet(Long projetId, Long projetDetailId){
        Projet projet = projetRepository.findById(projetId).get();
        ProjetDetail projetDetail = projetDetailRepository.findById(projetDetailId).get();
        // on set le fils dans le parent :
        projet.setProjetDetail(projetDetail);
        projetRepository.save(projet);
    }
```

Controller :

#http://localhost:8089/tp8/projet/affecter-projet-a-projet-details/**1/1**

```
    @PutMapping("/affecter-projet-a-projet-details/{projet-id}/{projet-details-id}")
    public void affecgterProjetAProjetDetail(@PathVariable("projet-id") Long proejtId,
    @PathVariable("projet-details-id") Long proejtDetailsId) {
        projetService.assignProjetDetailToProjet(proejtId, proejtDetailsId);
    }
```

Cas 3 : Affectation

On souhaite créer le service qui permet d'Affecter un Projet à une Equipe (le Projet et l'Equipe sont déjà créés dan la base de données) :

Solution :
Dans la méthode, on doit récupérer de la base de données l'Equipe et le Projet, puis affecter le Projet (fils) à l'Equipe (parent), puis faire une mise à jour :

Cas 3 : Affectation

Service :

```
public void assignProjetToEquipe(Long projetId, Long equipeId) {
        Projet projet = projetRepository.findById(projetId).get();
        Equipe equipe = equipeRepository.findById(equipeId).get();
        // on set le fils dans le parent  :
        equipe.getProjets().add(projet);
        equipeRepository.save(equipe);
    }
```

Attention : Initialiser la liste des Projets dans Equipe.java sinon erreur :

```
@ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private Set<Projet> projets = new HashSet<Projet>();
```

Or

```
@ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    private List<Projet> projets = new ArrayList<Projet>();
```

Controller (presque le même code que le Controller de l'Affectation 2) :
http://localhost:8089/tp8/projet/affecter-projet-equipe/**1/1**

Cas 4 : Affectation

On souhaite créer le service qui permet d'Ajouter un Projet, d'Affecter un Projet Detail à ce projet (Projet Détail est déjà dans la base de donnes) :

Solution :
Dans la méthode, on doit récupérer de la base de données le Projet Detail, puis le setter dans la le Projet (parent), puis faire une mise à jour :

Cas 4 : Affectation

Service :

```
public Projet addProjetAndAssignProjetToProjetDetail(Projet projet, Long projetDetailId) {
      ProjetDetail projetDetail = projetDetailRepository.findById(projetDetailId).get();
      // on set le fils dans le parent :
      projet.setProjetDetail(projetDetail);
      return projetRepository.save(projet);
   }
```

Controller à finir pour avoir l'URL :
http://localhost:8089/tp8/projet/creer-projet-et-affecter-projet-detail-a-projet/1
(le projet sera envoyé dans le body, l'id du projet detail dans le path).

Cas 5 : DésAffectation

On souhaite créer le service qui permet de désaffecter un ProjetDetail d'un Projet (Le Projet et le ProjetDetail sont déjà créés dans la base de données) :

Solution :
Récupérer le Projet de la base de données. Puis, il suffit de setter l'attribut projetDetail à **null** dans Projet, puis faire une mise à jour :

Cas 5 : DésAffectation

Service :

```
public Projet DesaffecterProjetDetailFromProjet(Long projetId) {
    Projet projet = projetRepository.findById(projetId).get();
    projet.setProjetDetail(null);
    return projetRepository.save(projet);
}
```

Controller à finir pour avoir l'URL :
http://localhost:8089/tp8/projet/desaffecter-projet-detail/1

Cas 5 : DésAffectation

Table t_projet_detail :

pd_id	pd_cout_provisoire	date_debut	pd_description	pd_technologie
16	0	2022-11-15	string	string

Table t_projet :

projet_id	projet_sujet	projet_detail_pd_id
2	string	

Cas 6 : DésAffectation

On souhaite créer le service qui permet de désaffecter un Projet d'une Equipe (L'Equipe et le Projet sont déjà créés dans la base de donnes) :

Solution :
Récupérer le Projet et l'Equipe de la base de données. Puis, il suffit d'enlever le projet à l'équipe, puis faire une mise à jour :

Cas 6 : DésAffectation

Service :

```
public void desaffecterProjetFromEquipe(Long projetId, Long equipeId) {
      Projet projet = projetRepository.findById(projetId).get();
      Equipe equipe = equipeRepository.findById(equipeId).get();
      // on enlève le fils du parent :
      equipe.getProjets().remove(projet);
      equipeRepository.save(equipe);
   }
```

Attention : Initialiser la liste des Projets dans Equipe.java sinon erreur :

```
@ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
   private Set<Projet> projets = new HashSet<Projet>();
```

Or

```
@ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
   private List<Projet> projets = new ArrayList<Projet>();
```

Controller à finir pour avoir l'URL :

http://localhost:8089/tp8/projet/desaffecter-projet-de-equipe/1/1

Important

Attention : Pour pouvoir faire les affectations et désaffectations ci-dessus, on doit injecter les repository nécessaires dans le service ProjetServiceImpl :

```
@Service
@AllArgsConstructor
   public class ProjetServiceImpl implements IProjetService {

      ProjetRepository projetRepository;

      ProjetDetailRepository projetDetailRepository;

      EquipeRepository equipeRepository;
```

TP8-1

Utiliser le projet **gestion-projet** développé en classe, pour développer et tester les **6 services affectations et désaffectations** vus dans le cours (voir slides ci-dessus).

TP8-2

Utiliser le projet **tp-foyer** développé en classe, pour développer et tester les **services affectations et désaffectations** suivants :

- Créer un Bloc et son Foyer en même temps.
- Affecter un Bloc à un Foyer (Les deux sont déjà dans la base de données).
- Désaffecter un Bloc de son Foyer

- Nous ajouterons d'autres exemples d'affectations désaffectations en classe.

Diagrammes de Classes