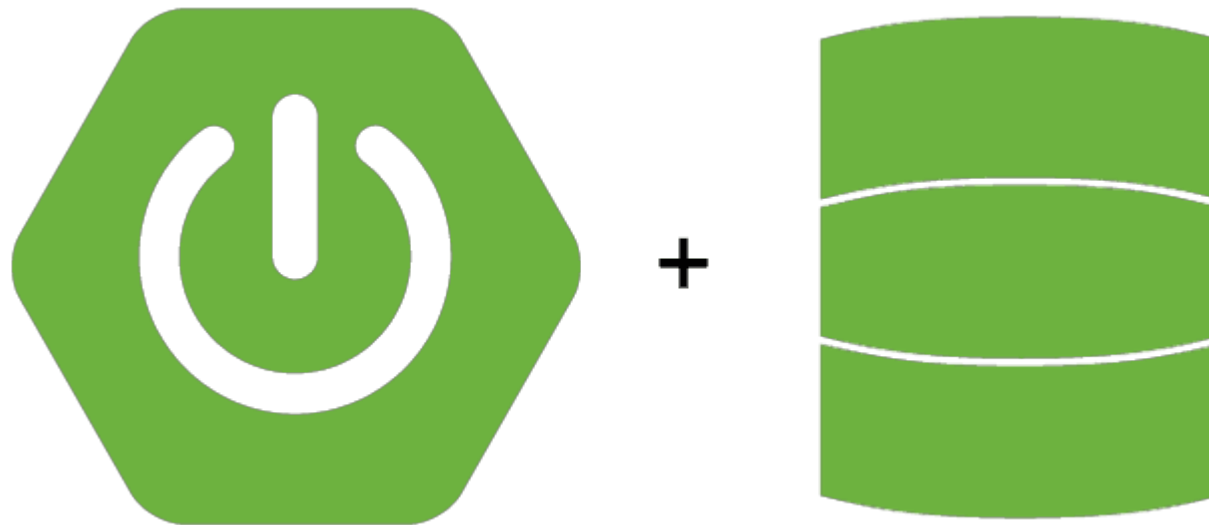


# SPRING DATA JPA - Fonctions avancées: Affectations



**UP ASI**  
**Bureau E204**

# Plan du Cours

- Règles de base
- Exemples pratiques
  - One To One
  - One To Many / Many To One
  - Many To Many

# Règles de base

- Pour réaliser les affectations d'une entité à une autre, il faut adopter une certaine démarche :

- ✓ S'il s'agit d'une relation **bidirectionnelle**, il faut distinguer le parent du child

Si la relation est **unidirectionnelle**, l'affectation se fera du côté de l'entité qui a accès aux attributs de l'autre (Parent)

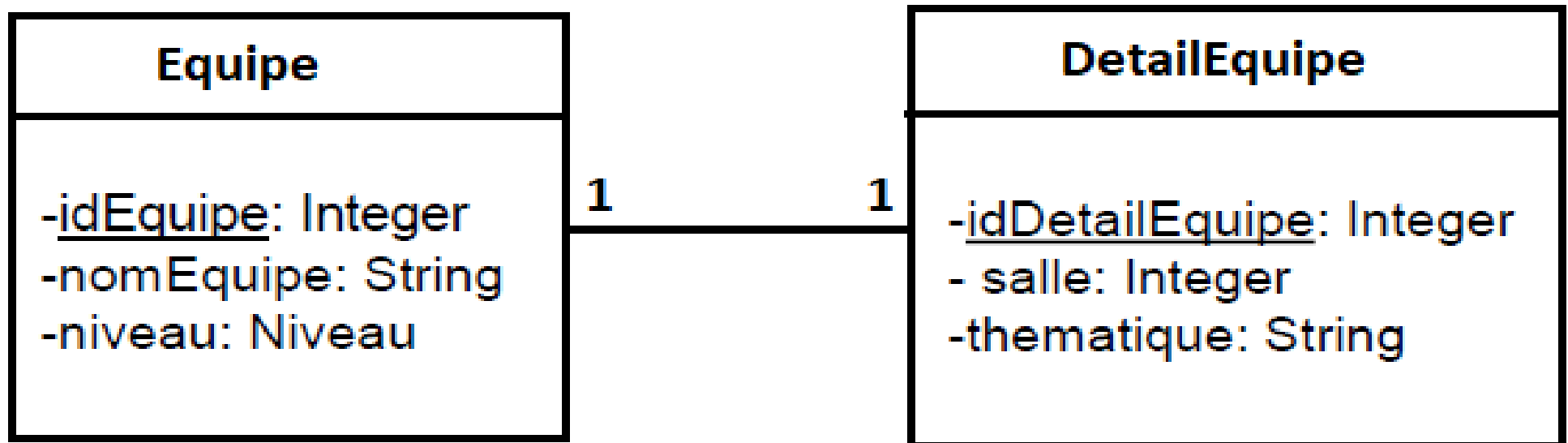
Les opérations d'affectation (setter) se réalise **exclusivement** sur l'entité parent.

□ **On affecte le child au parent.**

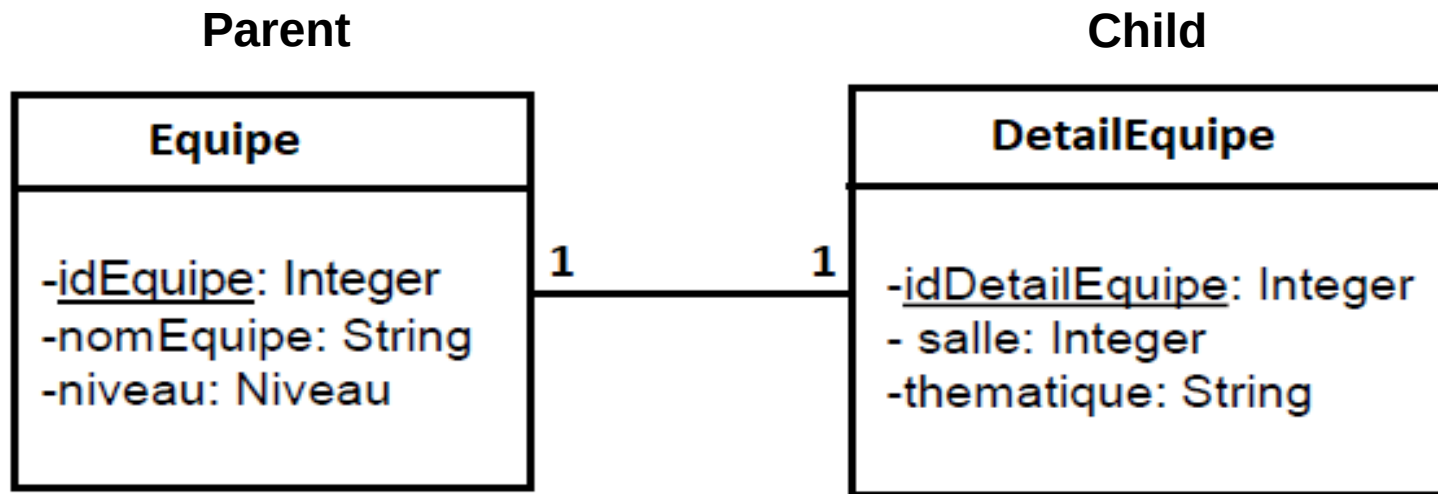
- ✓ Si vous faites l'affectation sur le child (fils), vous n'aurez pas d'erreur côté code mais **aucune modification** ne se réalisera côté base de données
- ✓ L'affectation est différente selon le type d'association (OneToOne, ManyToOne, OneToMany, ManyToMany) comme illustré dans les diapos suivants

# One To One

- Dans cet exemple, nous souhaitons **ajouter l'équipe, son détail** et **affecter ce DetailEquipe à l'Equipe** associée **en utilisant les cascades**.
- Il suffira donc par la suite de sauvegarder un objet Equipe et ses différents attributs primitifs (nomEquipe, niveau) et objet (DetailEquipe)



# One To One



```
public class Equipe implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idEquipe;  
    private String nomEquipe;  
    @Enumerated(EnumType.STRING)  
    private Niveau niveau;  
    @OneToOne(cascade = CascadeType.ALL)  
    private DetailEquipe detailEquipe;  
}
```

```
public class DetailEquipe implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idDetailEquipe;  
    private Integer salle;  
    private String thematique;  
    @OneToOne(mappedBy = "detailEquipe")  
    @JsonIgnore  
    private Equipe equipe;  
}
```

# One To One

## Ajout Equipe et Affectation à DetailEquipe

```
-  
public Equipe addEquipe(Equipe e) {  
  
    equipeRepository.save(e);  
    return e;  
}
```

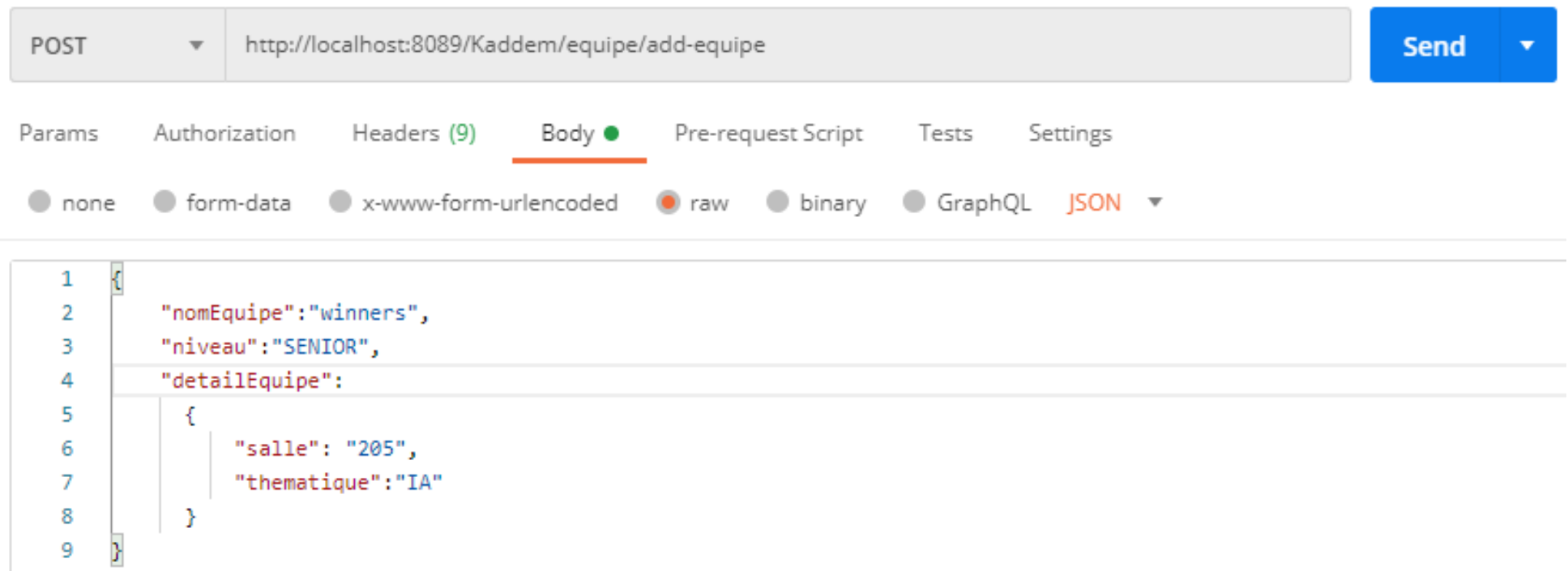
```
-  
public class Equipe implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idEquipe;  
    private String nomEquipe;  
    @Enumerated(EnumType.STRING)  
    private Niveau niveau;  
    @OneToOne(cascade = CascadeType.ALL)  
    private DetailEquipe detailEquipe;  
}
```

```
// http://localhost:8089/Kaddem/equipe/add-equipe  
/* cette méthode permet d'ajouter une équipe avec son détail*/  
@PostMapping("/add-equipe")  
@ResponseBody  
public Equipe addEquipe(@RequestBody Equipe e) {  
    Equipe equipe = equipeService.addEquipe(e);  
    return equipe;  
}
```

# One To One

## Ajout Equipe et Affectation à DetailEquipe

En utilisant l'option **cascade** entre **Equipe** et **DetailEquipe**, **l'objet Equipe envoyé doit contenir le détail équipe associé**. A l'exécution, nous aurons une ligne dans la table équipe, une ligne dans la table detailEquipe et la clé étrangère du detailEquipe dans l'équipe mise à jour.







# One To One

## Ajout Equipe et Affectation à DetailEquipe

- Avant l'exécution de la méthode, les tables équipe et DetailEquipe sont vides.
- Ces lignes seront présentes dans la base de données après l'exécution

 ▼				id_detail_equipe	salle	thematique
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	1	205	IA

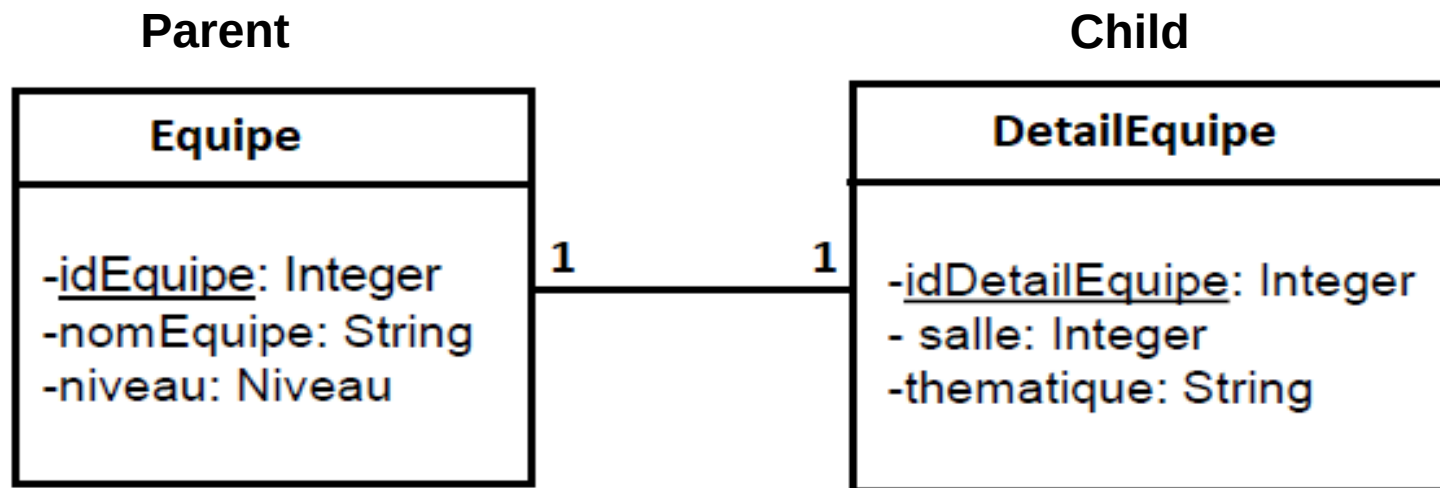
 ▼				id_equipe	niveau	nom_equipe	detail_equipe_id_detail_equipe
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	1	SENIOR	winners	1



# One To One

## Affectation DetailEquipe à Equipe

- Dans cet exemple, nous souhaitons **enregistrer le DetailEquipe** et **l'affecter à l'Equipe associée déjà existante dans la base.**
- **Equipe** est le parent donc l'affectation se fera sur l'objet **Equipe**



# One To One

## Affectation DetailEquipe à Equipe

```
public class Equipe implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idEquipe;  
    private String nomEquipe;  
    @Enumerated(EnumType.STRING)  
    private Niveau niveau;  
    @OneToOne  
    private DetailEquipe detailEquipe;  
}
```

```
public class DetailEquipe implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idDetailEquipe;  
    private Integer salle;  
    private String thematique;  
    @OneToOne(mappedBy = "detailEquipe")  
    @JsonIgnore  
    private Equipe equipe;  
}
```

- AffecterDetailsEquipeAEquipe(DetailEquipe d, int idEquipe) □ DetailEquipe: Child , Equipe: Parent
- 1/ Récupération des données : Equipe □ findById, DetailEquipe □ save
  - 2/ Child, Parent?
  - 3/ On affecte le child au parent : parent – un seul child (OneToOne) equipe.setDetailEquipe(de)
  - 4/ save du parent equipe.save(e)

# One To One

## Affectation DetailEquipe à Equipe

```
@Override
public Equipe affecterDetailEquipeToEquipe(DetailEquipe dt, Integer idEquipe) {
    Equipe equipe = equipeRepository.findById(idEquipe).get();
    // equipe parent dans l'association donc affecter le child au parent
    // sauvegarder l'objet detail equipe dans la bd
    DetailEquipe detailEquipe = detailEquipeRepository.save(dt);
    equipe.setDetailEquipe(detailEquipe);
    // sauvegarder le nouveau état de l'objet avec le detail affecté
    equipeRepository.save(equipe);
    return equipe;
}
```

# One To One

## Affectation DetailEquipe à Equipe

```
// http://localhost:8089/Kaddem/equipe/affecterDetailEquipeToEquipe/1
/* cette méthode permet d'affecter un détail équipe à son équipe */
@PutMapping("/affectDetailEquipeToEquipe/{idEquipe}")
@ResponseBody
public Equipe affecterDetailEquipeToEquipe(@RequestBody DetailEquipe dt,
                                           @PathVariable("idEquipe") Integer idEquipe) {
    Equipe equipe = equipeService.affecterDetailEquipeToEquipe(dt, idEquipe);
    return equipe;
}
```

# One To One

## Affectation DetailEquipe à Equipe

POST

http://localhost:8089/Kaddem/equipe/affecDetailEquipeToEquipe/1

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

{

2

"salle": 205,

3

"thematique": "IA"





4

}





# One To One

## Affectation DetailEquipe à Equipe





Cette ligne est déjà présente dans la base de données

		id_equipe	niveau	nom_equipe	detail_equipe_id_detail_equipe
<input type="checkbox"/>  Modifier  Copier  Effacer		1	SENIOR	winners	NULL

Ces lignes seront présentes dans la base de données après l'exécution

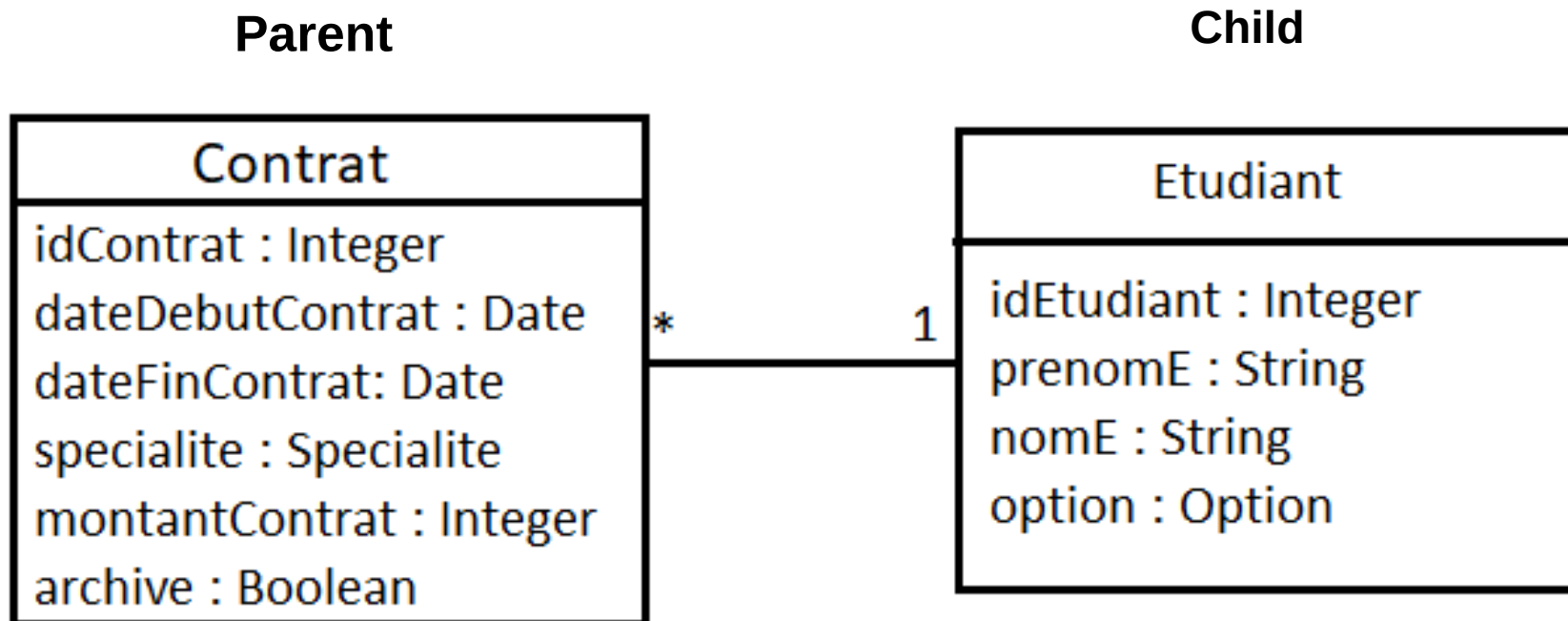
		id_detail_equipe	salle	thematique
<input type="checkbox"/>  Modifier  Copier  Effacer		1	205	IA

+ Options

		id_equipe	niveau	nom_equipe	detail_equipe_id_detail_equipe
<input type="checkbox"/>  Modifier  Copier  Effacer		1	SENIOR	winners	5

# One To Many

- Dans cet exemple, nous souhaitons **ajouter un étudiant avec les contrats associés**
- Il suffira donc par la suite de sauvegarder un objet Etudiant contenant une liste de contrats et affecter les étudiants (child) aux contrats (Parent).



# One To Many

## Ajout Etudiant avec les contrats associés

```
@Entity
public class Etudiant implements Serializable {
    private static final long serialVersionUID = 1L;
    4 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer idEtudiant;
    4 usages
    private String prenomE;
    4 usages
    private String nomE;
    4 usages
    @Enumerated(EnumType.STRING)
    private Option op;
    4 usages
    @OneToMany(mappedBy = "etudiant", cascade = CascadeType.ALL)
    // @JsonIgnore
    private List<Contrat> contrats;
```

```
@Entity
public class Contrat implements Serializable {

    private static final long serialVersionUID = 1L;
    4 usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer idContrat;
    4 usages
    @Temporal(TemporalType.DATE)
    private Date dateDebutContrat;
    4 usages
    @Temporal(TemporalType.DATE)
    private Date dateFinContrat;
    4 usages
    @Enumerated(EnumType.STRING)
    private Specialite specialite;
    4 usages
    private Boolean archived;
    4 usages
    private Integer montantContrat;
    4 usages
    @ManyToOne
    private Etudiant etudiant;
```



# One To Many

## Ajout Etudiant avec les contrats associés

```
ajouterEtudiantEtAffecterContrats(Etudiant e)
1/ récupération des données
List<Contrat> con= e.getContrats()
e = etudiantRepo.save(e)
2/ Child: Etudiant Parent: Contrat
3/ Appliquer la règle
for(Contrat c: con){
c.setEtudiant(e)
Contratrepo.save(c)
}
```

# One To Many

## Ajout Etudiant avec les contrats associés

```
// http://localhost:8089/Kaddem/etudiant/addEtudiantEtContratAssocie  
@PostMapping("/addEtudiantEtContratAssocie")  
@ResponseBody  
public Etudiant addEtudiantEtContratAssocie(@RequestBody Etudiant e) {  
    Etudiant etudiant= etudiantService.addEtudiantEtContratAssocie(e);  
    return etudiant;  
}
```

# One To Many

## Ajout Etudiant avec les contrats associés

POST

http://localhost:8089/Kaddem/etudiant/addEtudiantEtContratAssocie

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2     "prenomE": "ahmed",
3     "nomE" : "Sallem",
4     "op": "INFINI",
5     "contrats" : [
6         {
7             "dateDebutContrat": "2021-01-01",
8             "dateFinContrat": "2021-12-31",
9             "specialite": "CLOUD",
10            "montantContrat": 1000
11        }, {
12            "dateDebutContrat": "2022-01-01",
13            "dateFinContrat": "2021-12-31",
14            "specialite": "CLOUD",
15            "montantContrat" : 1200
16        }
17    ]
18 }
```

# One To Many

## Ajout Etudiant avec les contrats associés

Avant l'exécution, la base de données est vide

Après l'exécution, nous avons une ligne Etudiant dans la table Etudiant , deux lignes contrats et le champs etudiantId mis à jour avec l'id de l'étudiant ajouté

<div>←T→</div>				id_etudiant	nome	op	prenome	departement_id_departement
<div><div><input type="checkbox"/></div><div> Modifier</div></div>	<div><div> Copier</div><div> Effacer</div></div>	4	Sallem	INFINI	ahmed	NULL		

←T→ ▼				id_contrat	archived	date_debut_contrat	date_fin_contrat	montant_contrat	specialite	etudiant_id_etudiant
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	5	NULL	2021-01-01	2021-12-31	1000	CLOUD	4
<input type="checkbox"/>	 Modifier	 Copier	 Effacer	6	NULL	2022-01-01	2021-12-31	1200	CLOUD	4

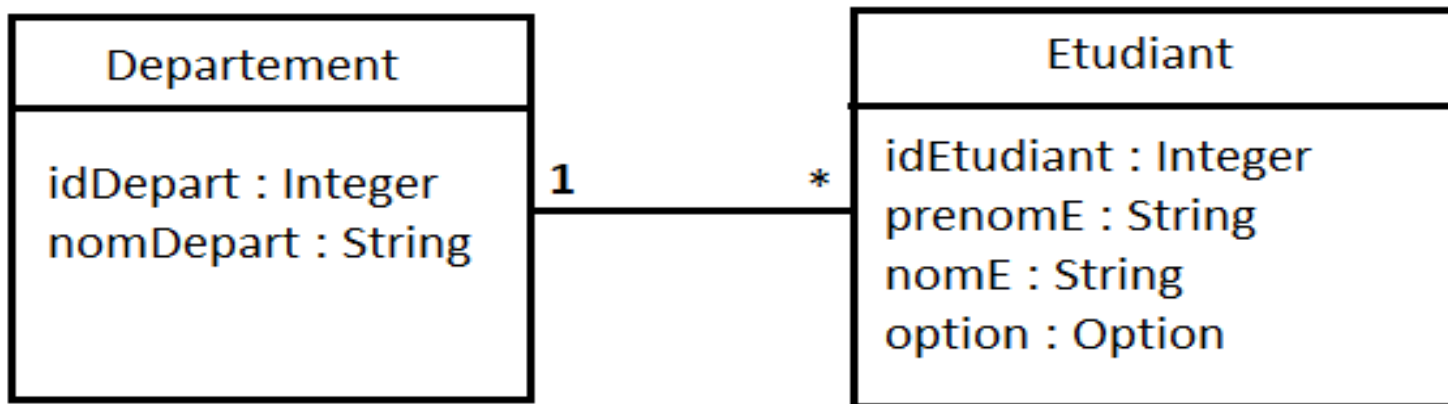
# One To Many

Dans cet exemple, nous souhaitons affecter un departement à l'Etudiant associé

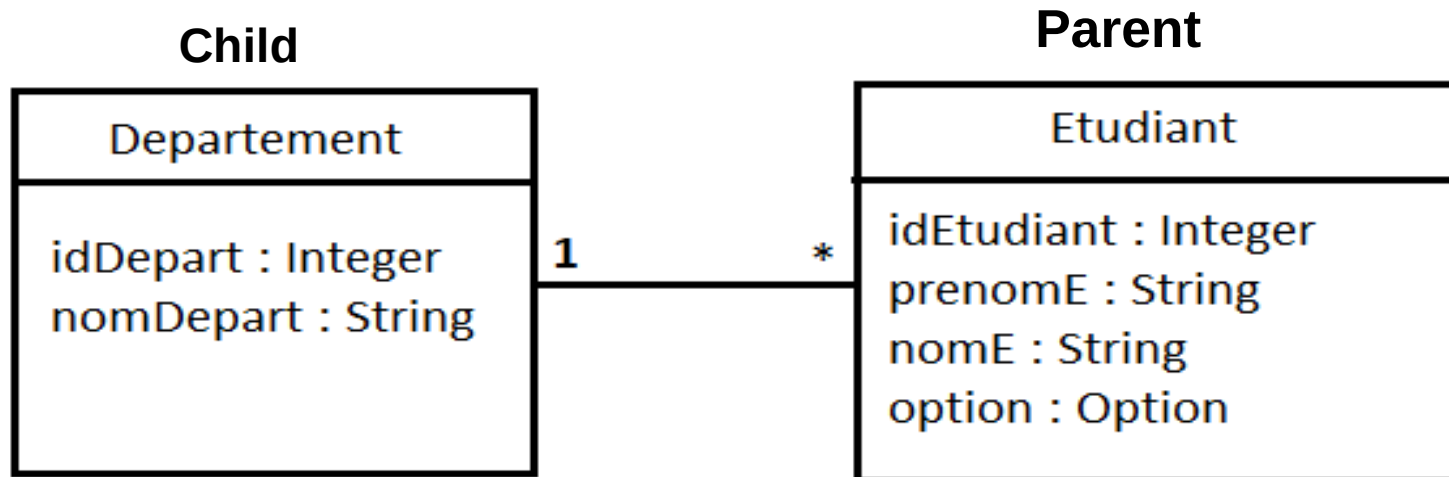
déjà présent dans la base de données

La cardinalité la plus forte est du côté de l'étudiant donc l'étudiant est le **parent**

L'affectation se fera **en dur** sur l'objet Etudiant



# One To Many



```
public class Departement implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idDepartement;  
    private String nomDepart;  
    @OneToMany(mappedBy = "departement")  
    @JsonIgnore  
    private List<Etudiant> etudiants;  
}
```

```
@Entity  
public class Etudiant implements Serializable {  
    private static final long serialVersionUID = 1L;  
    4 usages  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idEtudiant;  
    4 usages  
    private String prenomE;  
    4 usages  
    private String nomE;  
    4 usages  
    @Enumerated(EnumType.STRING)  
    private Option op;  
    4 usages  
    @ManyToOne  
    @JsonIgnore  
    private Departement departement;  
}
```

# One To Many

## Affectation Département à Etudiant

```
@Override
public void assignEtudiantToDepartement(Integer etudiantId, Integer departementId) {
    Etudiant e = etudiantRepository.findById(etudiantId).get();
    Departement d= departementRepository.findById(departementId).get();
    // affecter le child departement au parent Etudiant
    e.setDepartement(d);
    // sauvegarder le nouveau état de l'objet parent avec le département affecté
    etudiantRepository.save(e);
}
```

# One To Many

## Affectation Département à Etudiant

```
// http://localhost:8089/Kaddem/etudiant/assignEtudiantToDepartement/1/1
@PutMapping("/assignEtudiantToDepartement/{etudiantId}/{departementId}")
@ResponseBody
public void assignEtudiantToDepartement(@PathVariable("etudiantId") Integer etudiantId
    ,@PathVariable("departementId") Integer departementId) {
    etudiantService.assignEtudiantToDepartement(etudiantId,departementId);
}
```

PUT



http://localhost:8089/Kaddem/etudiant/assignEtudiantToDepartement/1/1

Send





# One To Many

## Affectation Département à Etudiant

Ces lignes sont déjà présentes dans la base de données

Options		id_etudiant	nome	op	prename	departement_id_departement
	Modifier  Copier  Effacer	1	slimi	INFINI	ahmed	NULL

Options		id_departement	nom_depart
	Modifier  Copier  Effacer	1	informatique

La ligne étudiant sera modifiée après l'exécution avec le département affecté à l'étudiant concerné

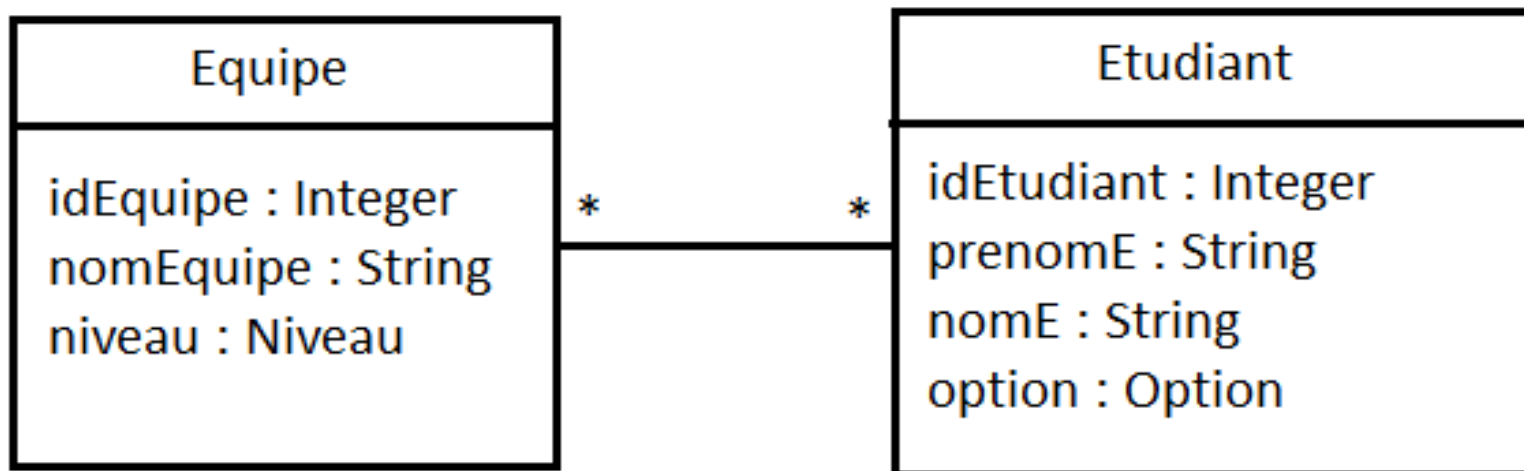
Options		id_etudiant	nome	op	prename	departement_id_departement
	Modifier  Copier  Effacer	1	slimi	INFINI	ahmed	1

# Many To Many

Dans cet exemple, nous souhaitons affecter une équipe à un Etudiant  
L'étudiant et l'équipe sont déjà sauvegardés dans la base de données

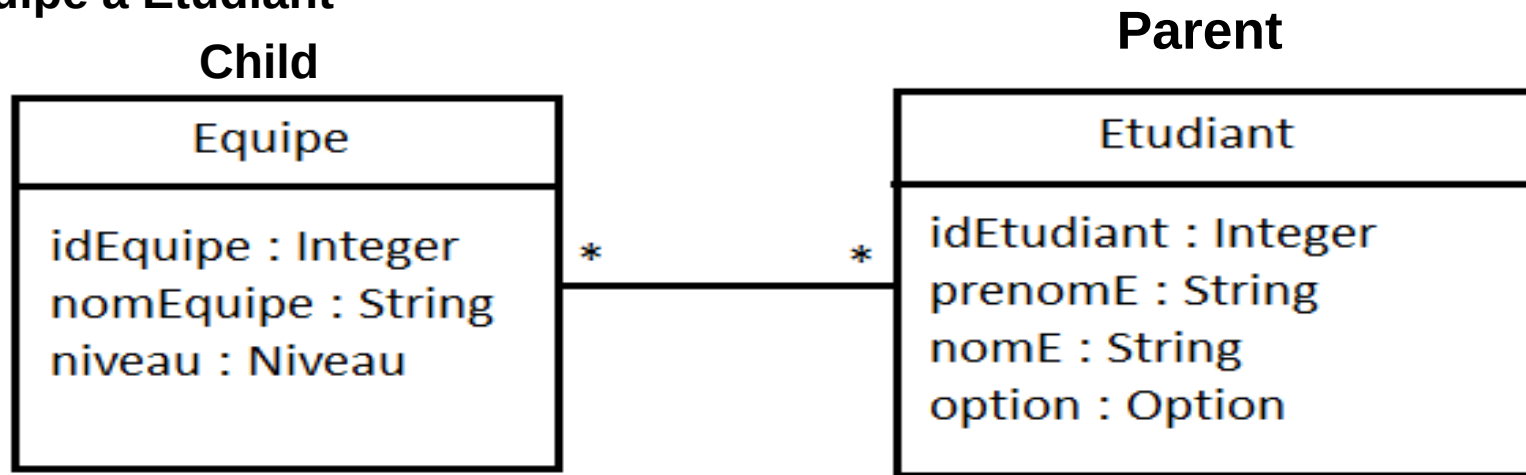
La cardinalité est la même des deux cotés de la relation donc c'est à nous de choisir le parent selon le besoin métier (ca sera **Etudiant** le parent dans ce cas)

L'affectation se fera sur l'objet **Etudiant**



# Many To Many

## Affectation Equipe à Etudiant



```
public class Equipe implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idEquipe;  
    private String nomEquipe;  
    @Enumerated(EnumType.STRING)  
    private Niveau niveau;  
    @ManyToMany(mappedBy = "equipes", cascade = CascadeType.ALL)  
    private List<Etudiant> etudiants;  
}
```

```
@Entity  
public class Etudiant implements Serializable {  
    private static final long serialVersionUID = 1L;  
    4 usages  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Integer idEtudiant;  
    4 usages  
    private String prenomE;  
    4 usages  
    private String nomE;  
    4 usages  
    @Enumerated(EnumType.STRING)  
    private Option op;  
    4 usages  
    @ManyToMany  
    @JsonIgnore  
    private List<Equipe> equipes;  
}
```

# Many To Many

## Affectation Equipe à Etudiant

```
@Override
public Etudiant affecterEquipeToEtudiant(Integer equipeId, Integer etudiantId) {
    Equipe equipe=equipeRepository.findById(equipeId).get();
    Etudiant etudiant = etudiantRepository.findById(etudiantId).get();
    // initialiser une liste d'équipe vide
    List<Equipe> equipesMisesAJour = new ArrayList<>();
    // récupérer les équipes déjà présentes dans la base
    if(etudiant.getEquipes()!=null) {
        equipesMisesAJour=etudiant.getEquipes();
    }
    // ajouter la nouvelle équipe à affecter
    equipesMisesAJour.add(equipe);
    // mettre à jour la liste des équipes
    etudiant.setEquipes(equipesMisesAJour);
    // Sauvegarder l'objet parent etudiant avec la liste des équipes mise à jour
    etudiantRepository.save(etudiant);
    return etudiant;
}
```







# Many To Many

## Affectation Equipe à Etudiant

```
// http://localhost:8089/Kaddem/etudiant/affecterEquipeToEtudiant/1/1
@PutMapping("/affecterEquipeToEtudiant/{equipeId}/{etudiantId}")
@ResponseBody
public void affecterEquipeToEtudiant(@PathVariable("equipeId") Integer equipeId,
                                     @PathVariable("etudiantId") Integer etudiantId) {
    etudiantService.affecterEquipeToEtudiant(equipeId, etudiantId);
}
```

# Many To Many

Ces lignes sont déjà présentes dans la base de données

	id_etudiant	nom	op	prenom	departement_id_departement
<input type="checkbox"/>  Modifier  Copier  Effacer	1	slimi	INFINI	ahmed	1
+ Options					
	id_equipe	niveau	nom_equipe	detail_equipe_id_detail_equipe	
<input type="checkbox"/>  Modifier  Copier  Effacer	1	SENIOR	winners	5	

Cette ligne sera présente dans la base de données après l'exécution avec l'équipe affecté à l'étudiant concerné dans la **table d'association** `etudiant_equipes`

t\_equipes

SELECT \* FROM `etudiant\_equipes`

Nombre de lignes : 25

+ Options

etudiants_id_etudiant	equipes_id_equipe
1	1

# **SPRING DATA JPA - Fonctions avancées Affectation**

Si vous avez des questions, n'hésitez pas à nous contacter :

**Département Informatique  
UP Architectures des Systèmes  
d'Information  
Bureau E204**