🧠 Spring Boot Service Layer — Cheat Sheet

Remplace **Entity** par ton entité, **entityRepo** par ton repository. Apprends les patterns, pas les noms.

1 GET ALL

```
public List<Entity> getAllEntities() {
    return this.entityRepo.findAll();
}
```

2 GET BY ID

```
public Entity getEntityById(Long id) {
    return
this.entityRepo.findById(id).orElse(null);
}
```

3 ADD — simple (pas de relation)

```
public Entity addEntity(Entity entity) {
    if (entity != null) {
        return this.entityRepo.save(entity);
    }
    return null;
}
```

4 ADD — avec @ManyToOne (parent par ID)

```
public Entity addEntity(Entity e, Long parentId)
{
    if (e == null) return null;
    Parent p =
this.parentService.getParentById(parentId);
    if (p == null) return null;
    e.setParent(p);
    return this.entityRepo.save(e);
}
```

5 ADD — avec @OneToMany / @ManyToMany (liste d'IDs)

```
public Entity addEntity(Entity e, List<Long>
childIds) {
    List<Child> children = childIds.stream()
        .map(id -> childService.getChildById(id))
        .toList();
    e.setChildren(children);
    return this.entityRepo.save(e);
}
```

6 UPDATE — full (trouver, copier, sauver)

```
public Entity updateEntity(Entity entity, Long
id) {
    Entity e =
this.entityRepo.findById(id).orElse(null);
    if (e == null)
        return this.entityRepo.save(entity); //
créer
    e.setField1(entity.getField1());
    e.setField2(entity.getField2());
    // ... chaque champ
    return this.entityRepo.save(e);
}
```

7 UPDATE — partiel (quelques champs)

```
public Entity updateFields(Long id, String f1,
String f2) {
    Entity e =
this.entityRepo.findById(id).orElse(null);
    if (e == null) return null;
    if (f1 != null) e.setField1(f1);
    if (f2 != null) e.setField2(f2);
    return this.entityRepo.save(e);
}
```

8 UPDATE — replace (setId + save)

```
public Entity updateEntity(Entity entity, Long
id) {
    if (entity == null ||

!this.entityRepo.findById(id).isPresent())
        return null;
    entity.setId(id);
    return this.entityRepo.save(entity);
}
```

9 DELETE — void

```
public void deleteEntityById(Long id) {
    this.entityRepo.deleteById(id);
}
```

10 DELETE — retourner l'objet supprimé

```
public Entity deleteEntity(Long id) {
    Entity e =
this.entityRepo.findById(id).orElse(null);
    if (e == null) return null;
    this.entityRepo.deleteById(id);
    return e;
}
```

11 ADD — avec vérif doublon (clé naturelle / String)

```
public Entity addEntity(Entity entity) {
    if
(this.entityRepo.findById(entity.getCode())
        .isPresent()) {
        return null; // existe déjà
    }
    return this.entityRepo.save(entity);
}
```

12 ASSIGN — ajouter enfant à parent

```
public Parent assignChildToParent(Child c, Long
pId) {
    Parent p =
this.parentRepo.findById(pId).orElse(null);
    if (p == null) return null;
    p.getChildren().add(c);
    return this.parentRepo.save(p);
}
```

13 FIND — entités liées via query custom

```
public List<Related> getRelatedByEntity(String
eId) {
    Entity e =
this.entityRepo.findById(eId).orElse(null);
    if (e == null) return null;
    return this.relatedRepo.findByEntity(e);
}
```

⚡ Référence rapide

Pattern	Return	Ligne clé
getAll	List<E>	`repo.findAll()`
getById	E	`repo.findById(id).orElse(null)`
add	E	`repo.save(entity)`
add + relation	E	set relation → `repo.save(entity)`
update full	E	find → copy fields → `repo.save(e)`
update partiel	E	find → if !=null set → `repo.save(e)`
update replace	E	`entity.setId(id)` → `repo.save(entity)`
delete void	void	`repo.deleteById(id)`
delete return	E	find → delete → return old
assign	Parent	find parent → `add(child)` → save parent

🏗️ Squelette de Service

```
@Service
public class EntityService {
    @Autowired
    private EntityRepo entityRepo;

    // coller les fonctions nécessaires ici
}
```

> 💡 **Règle d'or :** `this.repo.findById(id).orElse(null)` pour les lectures, `this.repo.save(obj)` pour les écritures. Tout le reste n'est que des variations autour de ces deux lignes.