Cascading

JPA

Topics Covered

- What is Cascading in JPA?
- Cascade Types and Their Usage
- Persist Cascade (CascadeType.PERSIST)
- Remove Cascade (CascadeType.REMOVE)

Cascading allows entity relationships to depend on the existence of another entity.

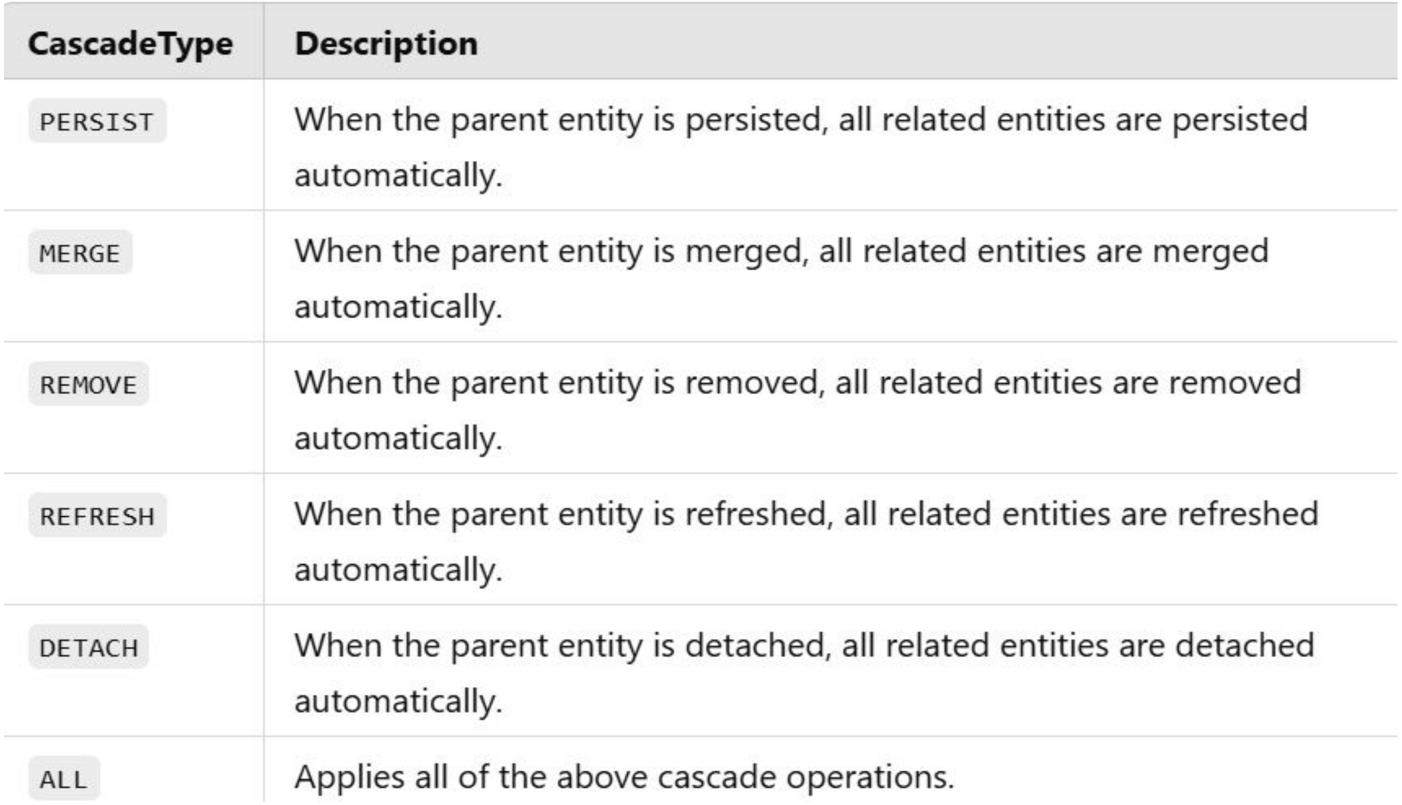
For example, in a Student-Tutor relationship, cascading helps in automatically persisting or removing related entities.

Cascade Types in JPA

JPA provides different cascade operations using the jakarta.persistence.CascadeType enum.

These cascade types can be applied to relationships like One-to-Many, Many-to-One, and Many-to-Many.

CascadeType Options



CascadeType	Description
`PERSIST`	When the parent entity is persisted, all related entities are persisted automatically.
`MERGE`	When the parent entity is merged, all related entities are merged automatically.
`REMOVE`	When the parent entity is removed, all related entities are removed automatically.
`REFRESH`	When the parent entity is refreshed, all related entities are refreshed automatically.
`DETACH`	When the parent entity is detached, all related entities are detached automatically.
`ALL`	Applies all of the above cascade operations.

Using CascadeType.PERSIST

Problem Without Cascading:

When saving a Tutor and their Students separately, missing a persist() call can lead to an error due to dependency.

Solution: Cascade Persist

With CascadeType.PERSIST, persisting the Tutor automatically persists all related Students.

Using CascadeType.PERSIST

Example - Tutor Class:

```
@OneToMany(cascade = CascadeType.PERSIST)
private Set<Student> teachingGroup;
```

Now, when a Tutor is persisted, all Students in teachingGroup are also saved automatically.

Using CascadeType.PERSIST

Example - Student Class (Reverse Cascade)

If we want students to trigger tutor persistence:

```
@ManyToOne(cascade = CascadeType.PERSIST)
@JoinColumn(name = "TUTOR_FK")
private Tutor tutor;
```

Now, persisting a Student ensures that their Tutor is also persist

Using CascadeType.REMOVE

If we want deleting a Tutor to remove all associated Students, we can use CascadeType.REMOVE.

Example - Tutor Class:

```
@OneToMany(cascade = {CascadeType.PERSIST, CascadeType.REMOVE})
private Set<Student> teachingGroup;
```

Now, deleting a Tutor removes all Students in the teachingGroup automatically

Summary

Feature	Description
CascadeType.PERSIST	Automatically persists related entities.
CascadeType.REMOVE	Automatically removes related entities.
Bidirectional Cascading	Can be applied in both parent and child entities.
Avoiding Errors	Helps prevent missing dependencies when persisting or removing entities.