1. CascadeType.PERSIST

Role: Propagates the persist operation from the parent entity to its associated child entities.

Use Case: If the parent entity is persisted, all associated child entities are also persisted.

Example:

java

Copy code

@OneToMany(mappedBy = "employee", cascade = CascadeType.PERSIST)

private List<Address> addresses;

When empRepo.save(employee) is called, the associated Address entities are persisted if not already in the database.

2. CascadeType.MERGE

Role: Propagates the merge operation from the parent entity to its associated child entities.

Use Case: If the parent entity is merged, all associated child entities are updated to reflect the current state.

Example:

java

Copy code

@OneToMany(mappedBy = "employee", cascade = CascadeType.MERGE)

private List<Address> addresses;

When empRepo.save(employee) is called with an existing employee, the associated Address entities are also merged (updated).

3. CascadeType.REMOVE

Role: Propagates the remove operation from the parent entity to its associated child entities.

Use Case: If the parent entity is deleted, all associated child entities are also deleted.

Example:

java

Copy code

@OneToMany(mappedBy = "employee", cascade = CascadeType.REMOVE)

private List<Address> addresses;

When empRepo.delete(employee) is called, all associated Address entities are deleted from the database.

4. CascadeType.REFRESH

Role: Propagates the refresh operation from the parent entity to its associated child entities.

Use Case: If the parent entity is refreshed (reloaded from the database), all associated child entities are also refreshed.

Example:

java

Copy code

@OneToMany(mappedBy = "employee", cascade = CascadeType.REFRESH)

private List<Address> addresses;

When entityManager.refresh(employee) is called, all associated Address entities are refreshed to their current state in the database.

5. CascadeType.DETACH

Role: Propagates the detach operation from the parent entity to its associated child entities.

Use Case: If the parent entity is detached (removed from the persistence context), all associated child entities are also detached.

Example:

java

Copy code

@OneToMany(mappedBy = "employee", cascade = CascadeType.DETACH)

private List<Address> addresses;

When entityManager.detach(employee) is called, all associated Address entities are detached from the persistence context.

6. CascadeType.ALL

Role: Combines all the above cascade types (PERSIST, MERGE, REMOVE, REFRESH, and DETACH).

Use Case: When you want all operations to be cascaded to the child entities.

Example:

java

Copy code

@OneToMany(mappedBy = "employee", cascade = CascadeType.ALL)

private List<Address> addresses;

Operations like save, update, delete, etc., performed on the Employee entity will also be propagated to the associated Address entities.

When to Use Each Cascade Type

| Cascade Type | Use Case |
| --- | --- |
| PERSIST | When you want associated entities to be automatically persisted along with the parent. |
| MERGE | When you need associated entities to be updated with the parent. |
| REMOVE | When deleting a parent entity, and you also want to delete associated child entities. |
| REFRESH | When you want child entities to stay in sync with the database when the parent is refreshed. |
| DETACH | When detaching a parent entity, and you want child entities to be detached as well. |
| ALL | When all operations on the parent should be cascaded to the child entities. |

Caution

Be cautious when using CascadeType.REMOVE or CascadeType.ALL. They may delete child entities unintentionally, which could lead to data loss.

Always evaluate whether a cascade type is necessary for your use case. Avoid blindly applying CascadeType.ALL without considering the impact.