Cascade

In JPA(Java Persistence API), if any operation is applied on an entity then it will perform on that particular entity only. These operations will not be applicable to the other entities that are related to it.

To establish a dependency between related entities, JPA provides **javax.persistence.CascadeType** enumerated types that define the cascade operations. These cascading operations can be defined with any type of mapping i.e. One-to-One, One-to-Many, Many-to-One, Many-to-Many.

If we don't provide any cascade type then no operations(Persist, Merge etc) would be in effect.

The main purpose of using JPA Cascade Types is that we can perform save or delete(or other operation) for Parent/Owner entity child/related entity will also be saved or deleted.

In JPA below Cascade Types have been defined.

- **CascadeType.PERSIST** – It means if the parent entity is saved then the related entity will also be saved.
- **CascadeType.MERGE** – It means if the parent entity is merged then the related entity will also be merged.
- **CascadeType.REMOVE** – It means if the parent entity is deleted then the related entity will also be deleted.
- **CascadeType.DETACH** – It means if the parent entity is detached then the related entity will also be detached.
- **CascadeType.REFRESH** – It means if the parent entity is refreshed then the related entity will also be refreshed.
- **CascadeType.ALL** – It is equivalent to cascade={DETACH, MERGE, PERSIST, REFRESH, REMOVE}.

Example on using JPACascade Types

We will use Cascade type persist for this example

Syntax:

@OneToOne(cascade=CascadeType.PERSISIT)

Example:

Book.java(Parent entity has OneToMany association with Story)

```
@Entity
public class Book {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	private int bookId;

	@Column(name = "book_name")
	private String bookName;

	@OneToMany(fetch= FetchType.LAZY)
	@JoinColumn(name = "book_id",referencedColumnName="bookId")
	private List<Story> storyList = new ArrayList<>();

}
```

Story.java(child entity)

```
@Entity
public class Story{
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
```

```
	private int id;
	@Column(name = "story_name")
	private String storyName;
}
```

We have below request data that we want to persist using postman.

```
{
	"bookName": "Premchand's best stories",
	"storyList": [{
			"storyName": "Stories of two oxes"
		},
		{
			"storyName": "idgah"
		},
		{
			"storyName": "Poosh Ki Rat"
		}
	]
}
```

We have BookServiceImpl.java as below.

```
@Service()
public class BookService {
```

```
    @Autowired
    private BookRepository bookRepository;

    @Transactional
    public Book saveBook(Book book) {
        book = bookRepository.save(book);
        return book;
    }

}
```

If we don't define cascade = CascadeType.PERSIST for OneToMany association(or any other association type) then we will not be able to persist the Story entity along with the Book entity. An exception will be thrown. In this case, we need to save the Story entity separately, which we might not want.

JPA GeneratedValue Annotation

The @GeneratedValue annotation provides the specification of generation strategies for the primary keys values.

Example:

@Id

@GeneratedValue(strategy = GenerationType.TABLE, generator = "student_generator")

Attributes:

1. Strategy:

The strategy attribute is used to specify the primary key generation strategy that the persistence provider must use to generate the annotated entity primary key. It is an optional attribute. Strategy values are defined in javax.persistence.GeneratorType enumeration which are as follows:

a. AUTO: Based on the database's support for the primary key generation framework decides which generator type to be used.

b. IDENTITY: In this case the database is responsible for determining and assigning the next primary key.

c. SEQUENCE: A sequence specifies a database object that can be used as a source of primary key values. It uses @SequenceGenerator.

d. TABLE: It keeps a separate table with the primary key values. It uses @TableGenerator.

Note: Default value of Strategy attribute is AUTO.

2. Generator:

The Generator attribute is used to specify the name of the primary key generator to use as specified in the SequenceGenerator or TableGenerator annotation. It is an optional attribute.