Key Points to Understand Associations in JPA and Hibernate

1. Entity Role:

- Every association involves two entities, each playing a role: **Owning Entity and Non Owning Entity.**
- The **owning** entity contains the **foreign key**, determining its role, while the other is the **non owning** or inverse side.

2. Cascade:

- **Cascade** defines how changes in the parent entity affect related child entities.
- JPA provides cascade types like **PERSIST, MERGE, REFRESH, REMOVE, DETACH, and ALL.**
- Example: **CascadeType.ALL** implies that operations on the **owning** entity also affect associated **non owning** entities.

3. Fetch Type:

- **FetchType** determines when data is fetched from the database—**EAGER or LAZY.**
- **EAGER** fetches all data in **one query**, while **LAZY** fetches data **on demand.**
- Example: **LAZY** loading retrieves only the essential data, loading additional data as needed.

4. Direction:

- Relationships can be **unidirectional or bidirectional.**
- **Bidirectional** relationships allow **navigational access in both directions**, enhancing query flexibility.

5. MappedBy Attribute:

The **mappedBy** attribute is used in **bidirectional relationships**, referring to the associated entities from **both sides**.

It helps establish the relationship correctly without duplicating foreign key mappings.

6. Join Column:

- **@JoinColumn** specifies a column for joining an entity association.
- The **owning entity** is identified by the presence of the **Join Column**, and it contains a **foreign key** referencing the **non owning** entity.

7. Join Table:

- **@JoinTable** is used in the mapping of associations and is specified on the owning side.
- It's applicable to **many to many** relationships and defines a separate table to store the relationship.
- Example: In a many to many relationship between **Engineering Branch** and **Subjects**, the join table (**BRANCH_SUBJECT**) holds **references** to **both tables**.