

JPA

Java Persistence API

Entity Types

- @OneToOne
 - One Entity related to one other
- @OneToMany
 - One entity is related to many entities (List, Set, Map, SortedSet, SortedMap)
- @ManyToOne
 - Inverse of OneToMany
- @ManyToMany
 - Many entities are related to many entities
 - Each has a List or Set reference to the other
 - A join table is used to define relationships
- Unidirectional - mapping is only done one way, one side does not know about relationship
- Bidirectional - both entities know about each other (recommended by hibernate)

The "Owning Side" of a relationship will hold the foreign key in the database.

Fetch Type

- Lazy Fetch Type - Data is not queried until referenced
- Eager Fetch Type - Data is queried up front
- Hibernate 5 supports the JPA 2.1 Fetch Type Defaults:
 - OneToMany - Lazy
 - ManyToOne - Eager
 - ManyToMany - Lazy
 - OneToOne - Eager

Cascade Types

If I delete the parent will the child be deleted as well?

- JPA Cascade Types control how state changes are cascaded from parent objects to child objects
- JPA Cascade Types
 - PERSIST - Save operations will cascade to related entities
 - MERGE - related entities are merged when the owning entity is merged
 - REFRESH - related entities are refreshed when the owning entity is refreshed
 - REMOVE - removes all related entities when the owning entity is deleted
 - DETACH - detaches all related entities if a manual detach occurs
 - ALL - Applies all the above cascade options.

By default, no operations are cascaded

Embeddable Types

- JPA/Hibernate support embeddable types
- These are used to define a common set of properties
- For example, a package with a shipping and billing address

Inheritance

- MappedSuperclass - Entities inherit from a super class. A database table IS NOT created for the super class.
- Single Table - (Hibernate Default) - One Table is used for all subclasses
 - This can lead to a lot of unused database columns
- Joined Table - Base class and subclasses have their own tables.
 - Fetching subclass entities require a join to the table of the superclass. (could cause performance issues)
- Table Per Class - Each subclass has its own table.

Create and Update Timestamps

- For audit purposes it is often a best practice to use timestamps
- JPA supports @PrePersist and @PreUpdate which can be used to support audit timestamps via JPA lifecycle callbacks.
- Hibernate provides @CreationTimestamp and @UpdateTimestamp