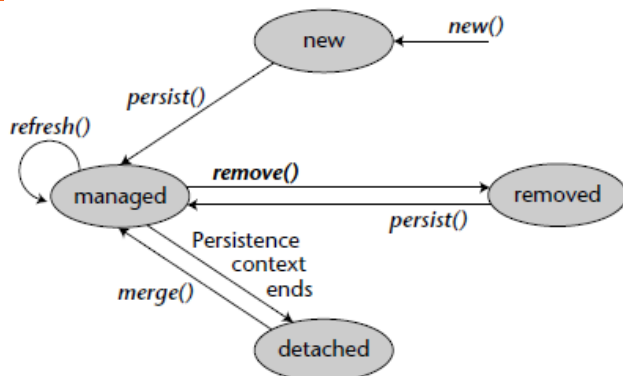


PersistenceContext

- PersistenceContext: connection between your in-memory instances and the database.
- PersistenceUnit: logical grouping of entity classes
- EntityManager: manipulates a PersistenceContext
- Scope TRANSACTION: context ends, when transaction ends (entities gets detached)
- Scope EXTENDED: context ends, when Stateful Bean gets removed. Entities may be managed over multiple transactions.
- Container-managed EntityManager: container opens or closes connection, access over @PersistenceContext
- @PersistenceContext (type=PersistenceContextType.TRANSACTION, unitName="intro")
EntityManager manager;
- Application-managed EntityManager: application opens or closes connection, access over EntityManagerFactory (or @PersistenceUnit)
- FlushMode defined on Query or EntityManager
- FlushMode COMMIT: db sync on commit
- FlushMode AUTO: db sync on commit and before queries
- Transaction in J2SE: EntityManager getTransaction()

Entity Class



Lifecycle of an Entity

- Can, but doesn't need to implement Serializable
- Public or Protected no-arg Constructor
- Can't be an inner class
- @Entity and needs a primary key (@Id)
- Annotation on field: Field access (reflection)
Annotation on getter: Property access
- Persist() doesn't perform a merge(), but merge() can perform a persist()!

Annotations

- @Table(name = "ACCOUNT_TABLE")
- @Column(name = "ACCOUNT_NO", nullable = false)
- @Transient
- @Temporal(TemporalType.DATE|TIME|TIMESTAMP)
- @SecondaryTable(name="DEPARTMENT",
pkJoinColumns={@PrimaryKeyJoinColumn(name="DEPT ID")})

Callbacks

- PrePersist
- PostPersist
- PreRemove
- PostRemove
- PreUpdate
- PostUpdate
- PostLoad (on find, getReference, refresh)
- Separate Listener: @EntityListeners(Test.class)
- exclude-superclass-listeners is possible

Locking

- Manager.lock(obj, {LockMode.WRITE|READ})
- Solves dirty read and nonrepeatable read
- READ: others can only read
- WRITE: others can only read and increment version (optimistic lock)
- @Version must be given for programmatic lock!

Lookup

- find: null, if not found
- getReference: EntityNotFoundException
- Query.getSingleResult() throws NonUniqueException or NoResultException, but will not rollback transaction

Relationship

- OneToOne
- OneToMany (with joinTable)
- ManyToOne (for backwards reference)
- ManyToMany (with joinTable)

Maps

- @MapKey
- key is a specific property of the related entity
- the value is the entity itself

Enumeration

- @Enumerated(EnumType.ORDINAL|STRING)

Cascade

- PERSIST
- MERGE
- REFRESH
- REMOVE