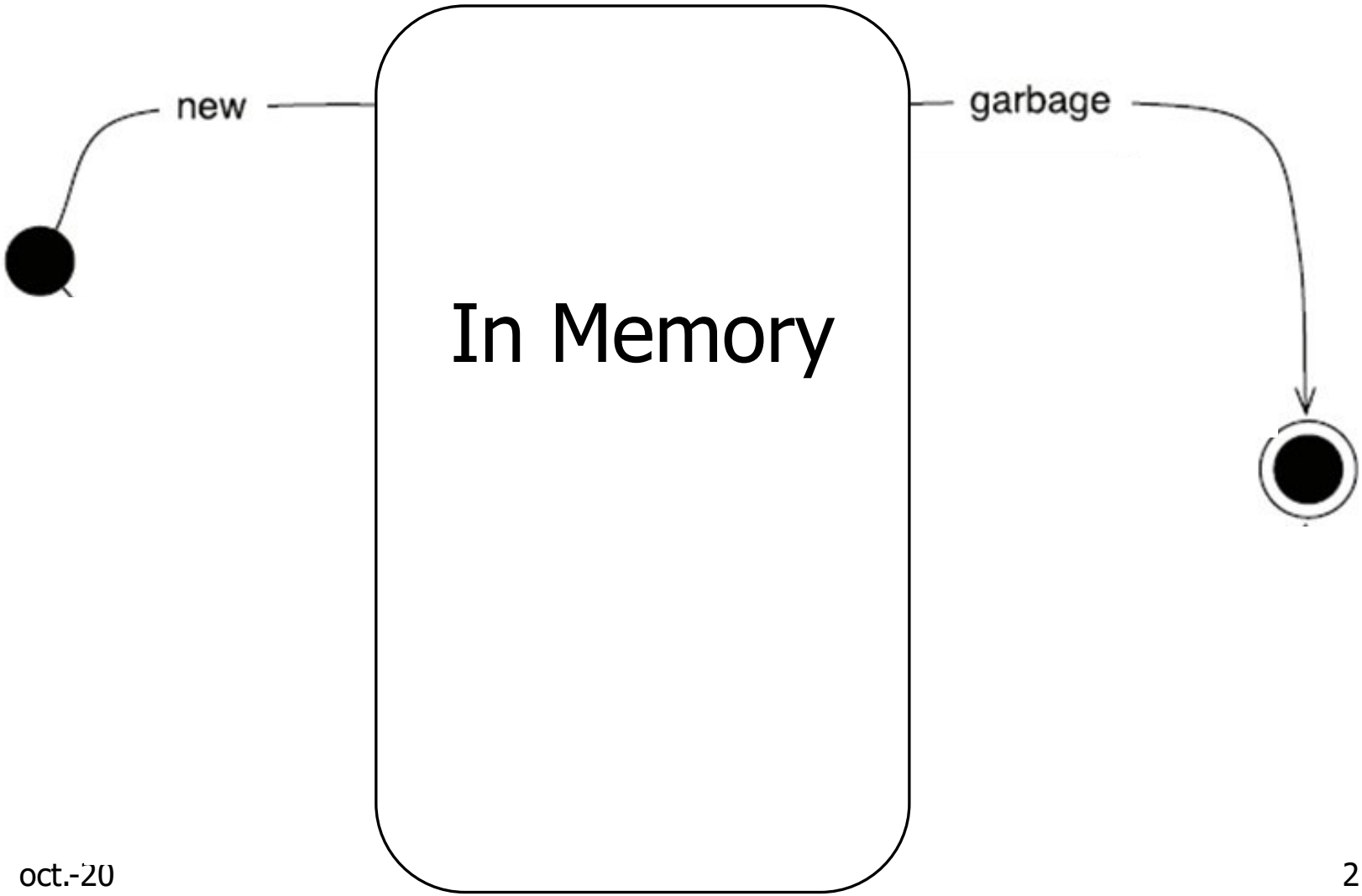
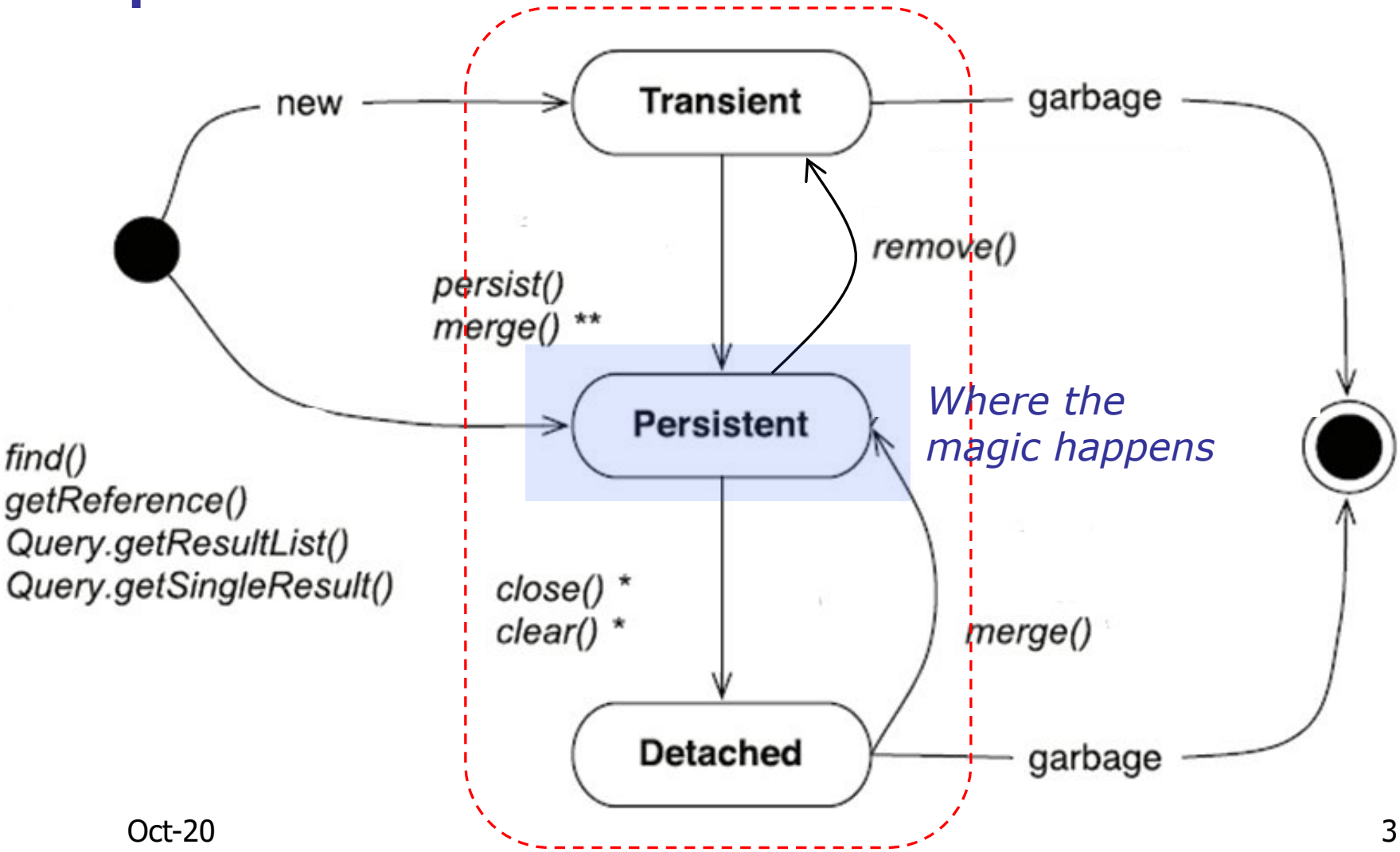
Gestión de objetos persistentes en JPA

Repositorios de Información

Ciclo de vida un objeto Java



Ciclo de vida de un objeto persistente



Estados de persistencia

- **Transient**
  - Un objeto recién creado que no ha sido enlazado con el gestor de persistencia (sólo existe en memoria de la JVM)
- **Persistent**
  - Un objeto enlazado con la sesión
  - Todos los cambios que se le hagan serán persistentes
  - La navegación por referencias carga el grafo
- **Detached**
  - Un objeto persistente que sigue en memoria después de que termina la sesión: existe en java y en la BDD

Control del ciclo de vida

- Se gestiona desde un EntityManager
  - Es el gestor de persistencia de JPA
- El EntityManager (la sesión) es el ámbito de persistencia
  - El ciclo de vida tiene lugar en la memoria de la JVM
  - Un objeto "está en sesión" cuando está en Persistent
- La sesión es una caché de primer nivel que:
  - Garantiza la identidad java y la identidad en BDD
    - No habrá varios objetos en sesión representando la misma fila
  - Se optimiza el SQL para minimizar tráfico a la BBDD
    - Dirty-checking
    - Write-behind

Dentro del contexto de persistencia(EntityManager) …

- Se lleva a cabo una unidad de trabajo (UoW)
- Al final de la unidad de trabajo se sincroniza con la BBDD
- La sesión lleva traza de todos los cambios hechos a los objetos en memoria durante la unidad de trabajo
- Al hacer COMMIT o FLUSH se organizan las actualizaciones para optimizar el rendimiento
- La identidad se garantiza porque una fila de la BBDD sólo se carga una vez y es representada por un único objeto java por contexto de persistencia
  - Pero puede haber muchos contextos simultáneos…

```
Category c = new Category();
c.setName("Gold"); // <-- Transient

EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
   em.persist( c ); // <-- Persistent
   c.setName("Golden");
tx.commit();
em.close();

c.setName("Golden Class"); // <-- Detached

EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
   em.merge( c ); // <-- Detached & DB updated
   c.setName("Golden"); // Detached
tx.commit();
em.close();
```

.../...

.../...

```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
   c = em.merge( c ); // <-- Persistent
   c.setName("First class");
tx.commit();
em.close();

EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
   c = em.merge( c ); // <-- Persistent
   em.remove( c ); // <-- Removed
tx.commit();
em.close();

c.setName("Good class"); //<-- Transient
```

Ámbito de identidad garantizada sólo dentro del contexto

```
EntityManager em1 = emf.createEntityManager();
EntityTransaction trx = em1.getTransaction();
trx.begin();

// Load Item with identifier value "1234"
Item a = em.find(Item.class, 1234);
Item b = em.find(Item.class, 1234);

assert( a == b ); // True, a and b are same object

trx.commit();
em1.close();

// References to a and b are now in detached state

EntityManager em2 = emf.createEntityManager();
EntityTransaction trx2 = em1.getTransaction();
trx2.begin();

Item c = em.find(Item.class, 1234);

assert( a == c ); // False, a and b are distinct objects

trx2.commit();
em2.close();
```

Navegación por el grafo

Navegar por zonas aún no cargadas del grafo es sólo posible con el contexto de persistencia abierto

Una vez cerrado sólo se puede navegar lo que ya está en memoria (detached)

Sincronización de la sesión y la BBDD (flush)

Ocurre los más tarde posible:

- Cuando se hace COMMIT a una transacción, o
- Antes de que se ejecute una consulta, o
- Cuando se llama `entityManager.flush()`

Se puede modificar el comportamiento

`entityManager.setFlushMode(…)`

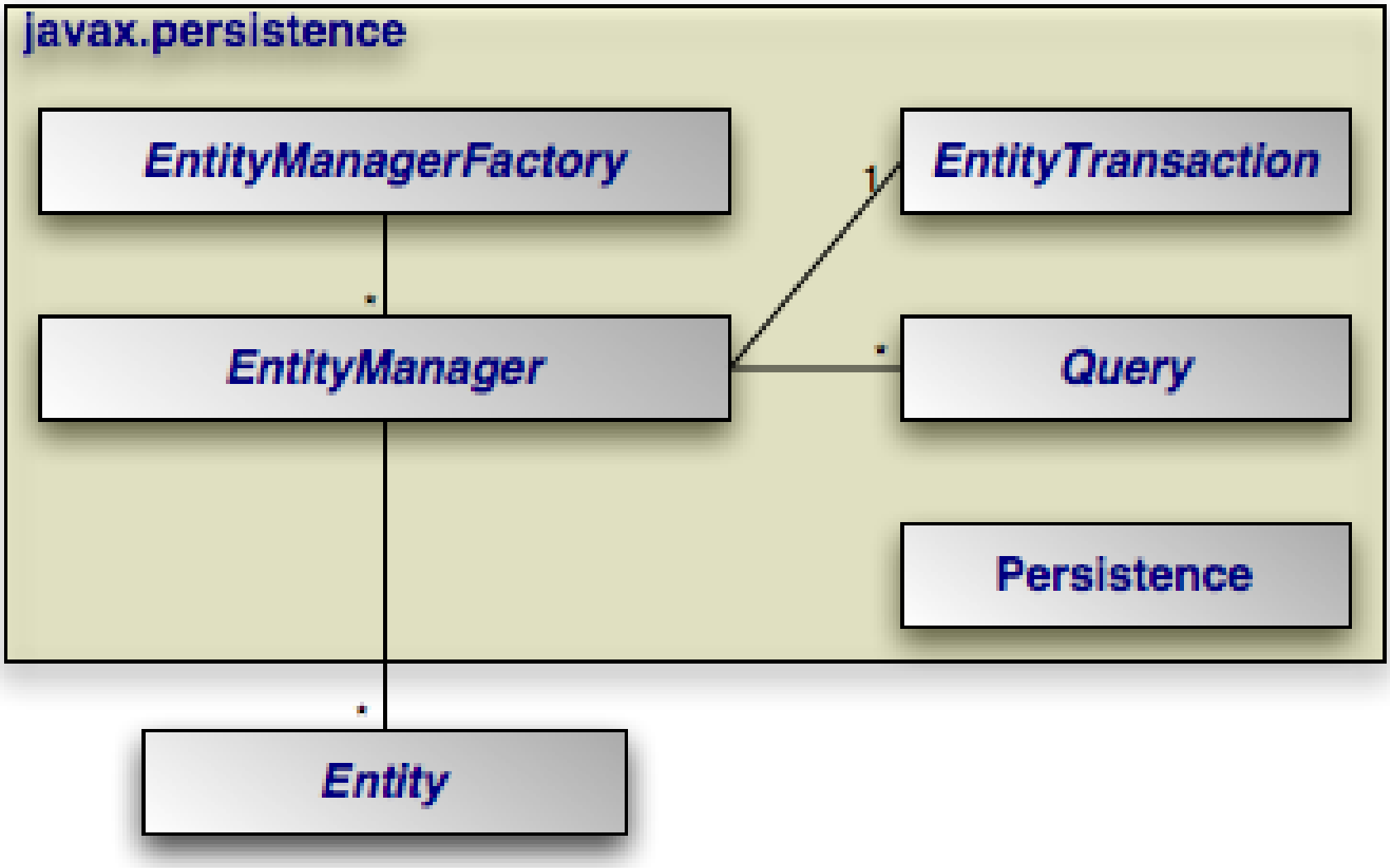
- `FlushMode.AUTO`
- `FlushMode.COMMIT`

Ámbito de persistencia y transacciones

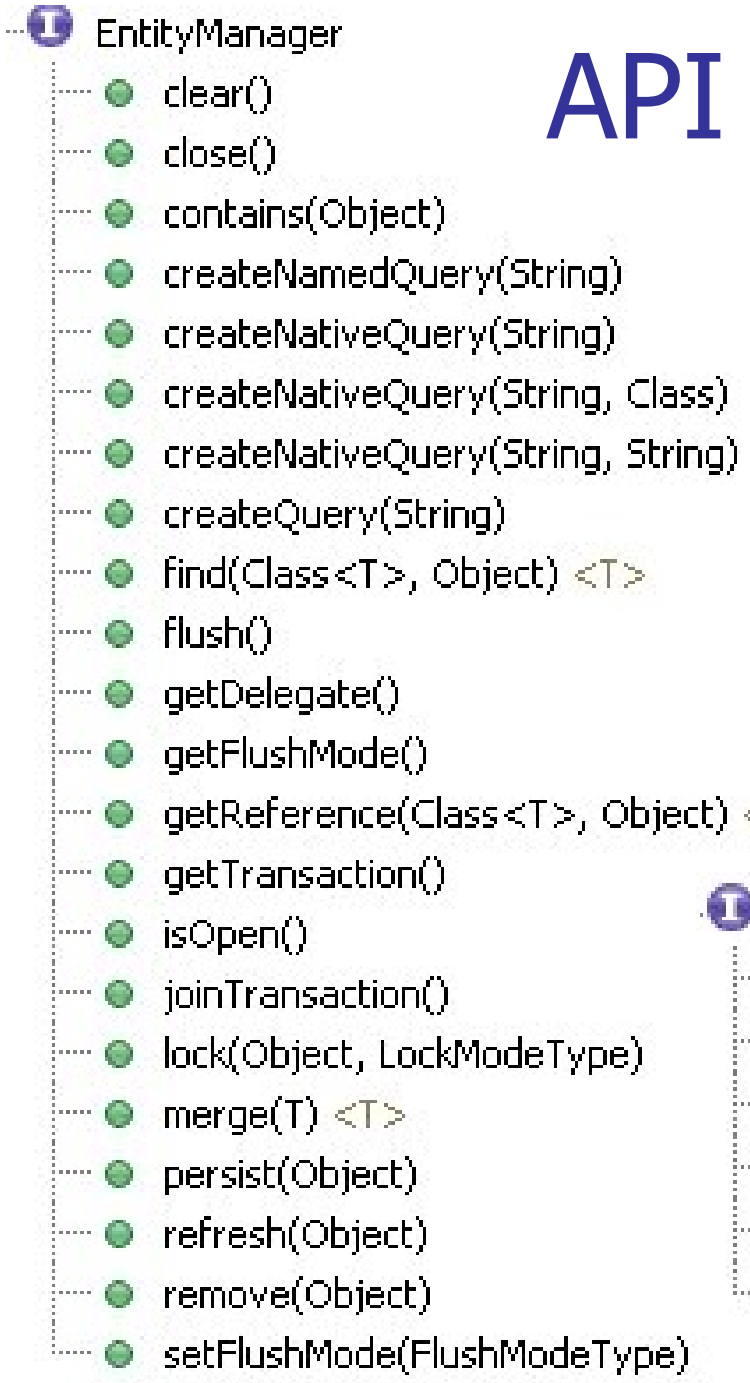
```
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();

tx.begin();
Item item = em.find(Item.class, new Long(1234));
tx.commit();

Item.setDescription(...);

tx.begin();
User user = em.find(User.class, new Long(3456));
user.setPassword("secret");
tx.commit();

em.close();
```

Item, todavía persistente, y user se salvan aquí

API JPA

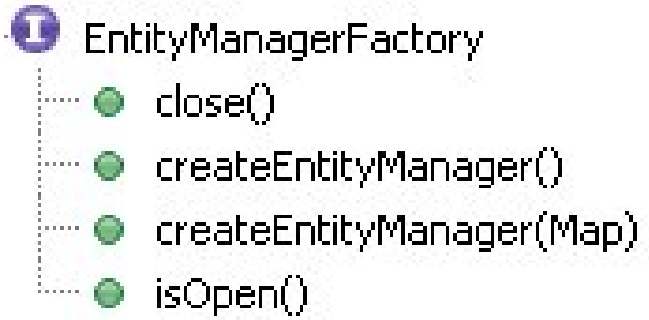

javax.persistence
EntityManagerFactory
EntityTransaction
1
EntityManager
Query
Persistence
Entity

API JPA


EntityManager
clear()
close()
contains(Object)
createNamedQuery(String)
createNativeQuery(String)
createNativeQuery(String, Class)
createNativeQuery(String, String)
createQuery(String)
find(Class<T>, Object) <T>
flush()
getDelegate()
getFlushMode()
getReference(Class<T>, Object) <T>
getTransaction()
isOpen()
joinTransaction()
lock(Object, LockModeType)
merge(T) <T>
persist(Object)
refresh(Object)
remove(Object)
setFlushMode(FlushModeType)


Query
executeUpdate()
getResultList()
getSingleResult()
setFirstResult(int)
setFlushMode(FlushModeType)
setHint(String, Object)
setMaxResults(int)
setParameter(int, Object)
setParameter(int, Calendar, TemporalType)
setParameter(int, Date, TemporalType)
setParameter(String, Object)
setParameter(String, Calendar, TemporalType)
setParameter(String, Date, TemporalType)


EntityTransaction
begin()
commit()
getRollbackOnly()
isActive()
rollback()
setRollbackOnly()


EntityManagerFactory
close()
createEntityManager()
createEntityManager(Map)
isOpen()

Gestionando objetos

- Inicio de la unidad de trabajo

```
EntityManagerFactory emf =
    Persistence.createEntityManagerFactory("caveatemptorDatabase");
EntityManager em = emf.createEntityManager();
EntityTransaction tx = em.getTransaction();
tx.begin();
```

- Fin de la unidad de trabajo

```
tx.commit();
em.close();
```

```
tx.rollback();
em.close();
```

Control de excepciones

```
EntityManager em = emf.createEntityManager();
try {
    EntityTransaction trx = em.getTransaction();
    trx.begin();
    try {
        // do business logic here
        ...
        trx.commit();
    } catch (PersistenceException ex) {
        trx.rollback();
        throw ex;
    }
}
finally {
    em.close();
}
```

Gestión más correcta del entity manager y transacción: control de las excepciones

Merge of a detached object: pseudocode

```
T merge(T obj) {
    T persistent;
    if ( context.contains( obj ) ) {
        persistent = obj;
    }
    else if ( bdd.contains( obj.id )) {
        persistent = context.load( obj.id );
        copyFromTo(obj, persistent);
    }
    else {
        persistent = context.persist( obj );
        copyFromTo(obj, persistent);
    }
    return persistent;
}
```

Si ya está en estado persistent

Si no es persistente, pero existe en BDD (detached)

Se carga de la BDD

Se copia a la versión persistente

If it is transient

Se añade al contexto de persistencia

Se devuelve la versión persistente