**What is Entity Manager?**

**Entity manager**. :

The **EntityManager** interface is an API that manages the lifecycle of an **entity** instance. In JPA, the **EntityManager** interface is used to allow applications to manage and search for **entities** in the relational database.

Entity Manager can be used in two ways:

* Container-managed entity manger: In this case, the instance of EntityManager is manged automatically by the container. We typically obtain the instance of EntityManager by injecting it with @PersistenceContext. This is only applicable to Java EE environment
* Application-managed entity manager: In this case, we have to explicitly obtain and manage the instance(s) of EntityManager. This is applicable to both Java EE or stand-alone/Java SE applications.

## What is Persistence Context?

* A persistence context is a set of entities such that for any persistent identity there is a unique entity instance.
* Within a persistence context, entities are managed.
* The EntityManager controls their lifecycle, and they can access datastore resource.

**What are types of Persistence Context?**

Transaction Persistence Context:

An EntityManager begins a new persistence context with each transaction, and ends the context when the transaction commits or rolls back.

ex:

EntityManager em; // injected

...

// outside a transaction:

// each operation occurs in a separate persistence context, and returns

// a new detached instance

Magazine mag1 = em.find(Magazine.class, magId);

Magazine mag2 = em.find(Magazine.class, magId);

assertTrue(mag2 != mag1);

...

// transaction begins:

// within a transaction, a subsequent lookup doesn't return any of the

// detached objects. however, two lookups within the same transaction

// return the same instance, because the persistence context spans the

// transaction

Magazine mag3 = em.find(Magazine.class, magId);

assertTrue(mag3 != mag1 && mag3 != mag2);

Magazine mag4 = em.find(Magazine.class (magId);

assertTrue(mag4 == mag3);

...

// transaction commits:

// once again, each operation returns a new instance

Magazine mag5 = em.find(Magazine.class, magId);

assertTrue(mag5 != mag3);

**What is Persistence Unit?**

A persistence unit is a logical grouping that contains information like configuration of EntityManagerFactory, a set of entity classes, mapping metadata.

Each persistence-unit must have a unique name. An application can have one or more persistence units.

**Does entity manager have update method?**

No, as per given in javadoc.

load vs get:

**1. session.load()**

* It will always return a “proxy” (Hibernate term) without hitting the database. In Hibernate, proxy is an object with the given identifier value, its properties are not initialized yet, it just look like a temporary fake object.
* If no row found , it will throws an ObjectNotFoundException.

**2. session.get()**

* It always hit the database and return the real object, an object that represent the database row, not proxy.
* If no row found , it return null.

Extended Persistence Context:

Maintains the same persistence context for its entire lifecycle. Whether inside a transaction or not

EntityManagerFactory emf = ...

EntityManager em = emf.createEntityManager();

// persistence context active for entire life of EM, so only one entity

// for a given persistent identity

Magazine mag1 = em.find(Magazine.class, magId);

Magazine mag2 = em.find(Magazine.class, magId);

assertTrue(mag2 == mag1);

em.getTransaction().begin();

// same persistence context active within the transaction

Magazine mag3 = em.find(Magazine.class, magId);

assertTrue(mag3 == mag1);

Magazine mag4 = em.find(Magazine.class (magId);

assertTrue(mag4 == mag1);

em.getTransaction.commit ();

// when the transaction commits, instance still managed

Magazine mag5 = em.find(Magazine.class, magId);

assertTrue(mag5 == mag1);

// instance finally becomes detached when EM closes

em.close();