Es una API contenida dentro del JEE.

Provee la facilidad de mapeo de objeto/relación a los desarrolladores Java para el manejo en aplicaciones Java.

La persistencia se centra en tres areas:

La API para la persistencia

El lenguaje de consultas

Metadatos de mapeo objeto/relacional

Conceptos a manejar:

Entities

Entity Manager

Persistence unit

Son clases tipo POJO (Plain Old Java Object) que identifican el uso de clases simples y que no dependen de un framework en especial.

Son el punto intermedio al nivel de persistencia, que no siempre es una base de datos relacional.

De este modo, JEE prevé una estricta separación entre el nivel de datos y el lógico.

Representa con sus atributos las columnas de una tabla de un banco de datos.

Un desarrollador JEE no necesita tener un contacto directo con la base de datos, sino simplemente manejar los datos mediante instancias de Entidad

A diferencia de los Beans de sesión, no implementan ninguna interface especial.

Cuando no se manejan Anotaciones en estas clases, nada puede hacerle referencia a una clase de entidad.

Este tipo de instancias pueden usarse en cualquier parte, siempre es posible enviarlas al cliente.

A menudo son objetos serializables, de modo que su referencia puede ser retenida por Session Bean Stateful

Deben de disponer de un constructor estándar porque entre otros, es tarea del EntityManager crear instancias a partir de esto.

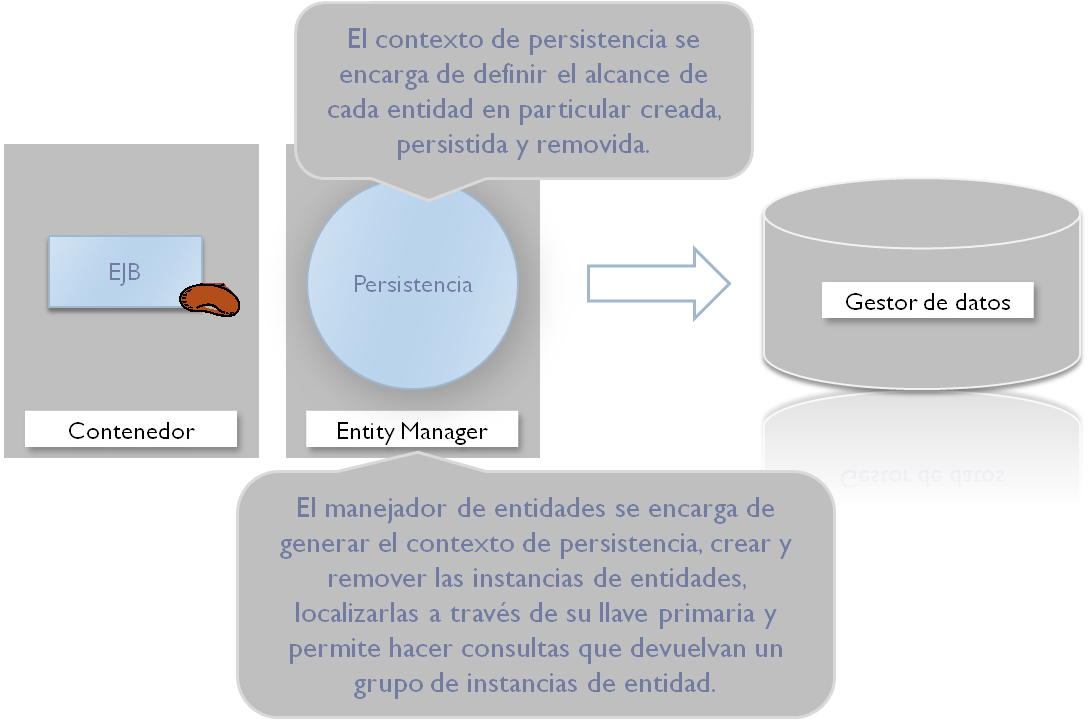
Para que una clase de entidad tenga sentido es necesario asociarla a un EntityManager.

* Requerimientos
  + Debe de ser anotada con javax.persistence.Entity
  + La clase debe de contener un constructor public/protected estándar (sin argumentos)
  + La clase no puede ser declarada como final, ni alguno de sus métodos o variables de instancia.
  + Si se transmitirá el objeto al cliente, entonces debe ser marcada como serializable.
  + Las clases de entidad pueden extender clases de entidad y no entidad, las clases de no entidad pueden extender clases de entidad.
  + Las variables de instancia persistente deben ser declaradas private, protected o default y pueden ser accedidas solo por métodos de la clase entidad. Los clientes deben de acceder el estado de persistencia a través de métodos de negocio.
* Campos persistentes:
  + Los tipos de datos permitidos para el manejo de campos persistentes son:
    - Tipos primitivos
    - String
    - Wrapper de tipos primitivos
    - BigInteger, BigDecimal
    - Date, Calendar
    - java.sql.Date, java.sql.Time, java,sql.Timestamp
    - Byte[], byte[], Character[], char[]
    - Los definidos por el usuario como serializables
    - Tipos enumerados
    - Otras entidades o Collection de entidades
    - Clases embebidas
* Todos los campos no Anotados como Transient serán persistentes en la base de datos.
* Podemos usar persistencia en las propiedades, para esto es necesario seguir la convención de componentes JavaBeans.
  + Pueden usar interfaces de Collections Java.
    - Collection
    - Set
    - List
    - Map
* Llaves primarias
  + El propósito de establecer llaves primarias dentro del objeto es hacerlo único dentro de la capa de persistencia.
  + Este identificador único permite a los clientes localizar una instancia del objeto entidad.
  + Para poder establecer un campo como llave primaria es necesario colocar la una Anotación javax.persistence.Id
  + Podremos tener una llave primaria compuesta por más de un campo, la forma de establecerla es mediante la creación de un objeto tipo llave que contenga los campos que formarán la llave primaria, De esta forma un solo campo y tipo representarán la llave primaria.

|  |
| --- |
| public final class LineItemKey implements Serializable {  public Integer orderId;  public int itemId;  public LineItemKey() {}  public LineItemKey(Integer orderId, int itemId) {  this.orderId = orderId;  this.itemId = itemId;  }  public boolean equals(Object otherOb) {  if (this == otherOb) {  return true;  }  if (!(otherOb instanceof LineItemKey)) {  return false;  }  LineItemKey other = (LineItemKey) otherOb;  return ((orderId==null?other.orderId==null:orderId.equals(other.orderId)) && (itemId == other.itemId));  }  public String toString() {  return "" + orderId + "-" + itemId;  }  } |

* Llaves primarias.
  + Una de las ventajas que nos da el manejo de llaves primarias es el poder establecer relaciones entre los objeto y así poder representar un entorno de persistencia equivalente a una base de datos relacional y mantener integridad en los datos.
  + Esto implica que el programador se olvidará de esta tarea de sistema, la cual consiste en mantener la integridad de la información a través de Constraints ubicados en la capa de datos o validaciones en la capa lógica.
  + Se pueden manejar cuatro tipo de relaciones en la capa de persistencia:
    - Uno a uno
    - Uno a muchos
    - Muchos a uno
    - Muchos a muchos
* Relaciones. Dirección
  + Existen dos tipos de dirección en la relación entre los objetos. La dirección establece la forma en que podrán viajar los datos entre los objetos en la capa de persistencia.
  + Bidireccional
    - En este tipo de direccionamiento ambas clases de entidad se conocen, es decir la entidad A hace referencia a la entidad B y así mismo B establece una referencia con A.
  + Unidireccional

Solo una entidad conoce a la otra, es decir, la clase entidad A contiene a la clase entidad B, sin embargo B desconoce la entidad A.



* Podemos manejar las instancias de entidad de dos formas:
  + Manejador de contenidos
    - Bajo este escenario, el contexto de entidad en la instancia del manejador de contenido es propagado automáticamente por el contenedor a todos los componentes de aplicación que usan la instancia de EntityManager dentro de una sola transacción JTA.
  + Manejador de aplicación
    - En este caso, el contexto de persistencia no es propagado y el ciclo de vida es manejado por la aplicación. Es utilizado este escenario para acceder a una unidad de persistencia que no es propagada con la transacción JTA. De esta forma, cada manejador de entidad EntityManager crea un nuevo contexto de persistencia aislado. Esta instancia se genera a través de un método de una instancia de factoria del EntityManager.

Localizando una entidad a través del EntityManager.

|  |
| --- |
| @PersistenceContext  EntityManager em;  public void enterOrder(int custID, Order newOrder) {  Customer cust = em.find(Customer.class, custID);  cust.getOrders().add(newOrder);  newOrder.setCustomer(cust);  } |

Instancias de entidad persistentes

|  |
| --- |
| @PersistenceContext  EntityManager em;  ...  public LineItem createLineItem(Order order, Product product, int quantity) {  LineItem li = new LineItem(order, product, quantity);  order.getLineItems().add(li);  em.persist(li);  return li;  } |

Removiendo una instancia de entidad.

|  |
| --- |
| @PersistenceContext  EntityManager em;  public void removeOrder(Integer orderId) {  try {  Order order = em.find(Order.class, orderId);  em.remove(order);  }... |

Creando consultas de instancias de entidad.

|  |
| --- |
| @PersistenceContext  EntityManager em;  public List findWithName(String name) {  return em.createQuery(  "SELECT c FROM Customer c WHERE c.name LIKE :custName")  .setParameter("custName", name)  .setMaxResults(10)  .getResultList();  } |

* La anotación @NamedQuery especifica el nombre de una consulta que será posteriormente usada, su equivalente en el gestor de datos vendría siendo un objeto Vista.
* Este elemento puede ser creado a través del método *createNamedQuery*

|  |
| --- |
| // Definición de la consulta en la clase entidad:  @NamedQuery(  name="findAllCustomersWithName",  query="SELECT c FROM Customer c WHERE c.name LIKE :custName"  )  // Así lo debe de generar el cliente que lo consume  @PersistenceContext  EntityManager em;  …  customers = em.createNamedQuery("findAllCustomersWithName").setParameter("custName", "Smith“).getResultList(); |

* La unidad de persistencia define una asignación de clases de entidad que serán controladas por el manejador de contenidos instanciado en la aplicación.
* Esta asignación representa los datos contenidos dentro de un solo repositorio de datos.
* Las unidades de persistencia se definen en un archivo de configuración nombrado *persistence.xml* que debe de ser colocado en el directorio META-INF de nuestro JAR.
* Cada unidad de persistencia es identificada con un nombre único dentro de su alcance.
* Cuando generes un JAR de EJB, el archivo de persistencia deberás ubicarlo en el directorio META-INF del Jar.
* Si generas un WAT, el archivo de persistencias deberá de colocarse en el directorio WEB-INF/classes/META-INF
* El archivo XML de persistencia puede definir una o más unidades de persistencia.
* Aquí un ejemplo de un archivo de configuración persistence.xml con dos clases de entidad que serán controladas por el manejador de entidades a través de la unidad de persistencia.

|  |
| --- |
| <persistence>  <persistence-unit name="OrderManagement">  <description>This unit manages orders and customers.</description>  <jta-data-source>jdbc/MyOrderDB</jta-data-source>  <jar-file>MyOrderApp.jar</jar-file>  <class>com.widgets.Order</class>  <class>com.widgets.Customer</class>  </persistence-unit>  </persistence> |