Trabajando con JPA

Entity Manager y contexto de persistencia

Índice del capítulo

Introducción

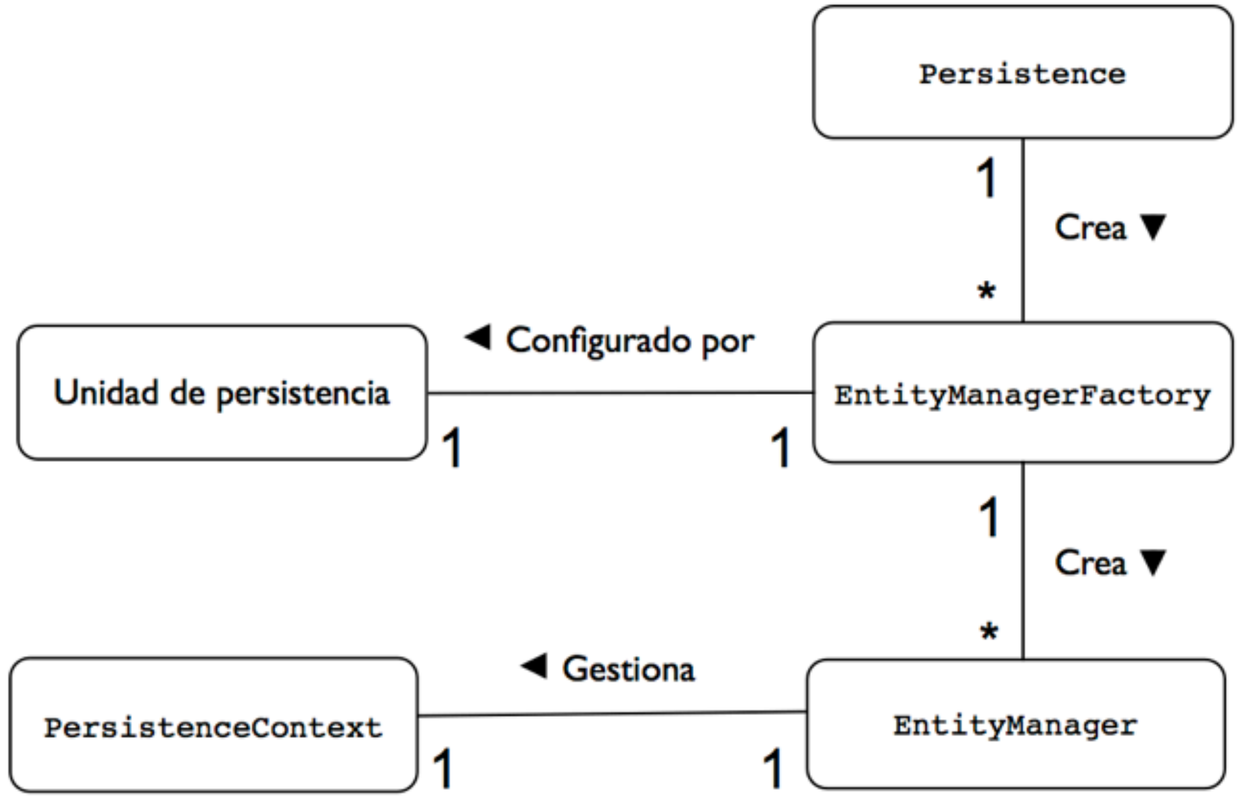
- **Todas las operaciones relacionadas con la persistencia** de las entidades se realizan a través de un **gestor de entidades** (entity manager en inglés).
- El funcionamiento del entity manager está especificado en una **única interfaz llamada EntityManager**.
- El entity manager tiene dos responsabilidad fundamentales:
  - Define una **conexión transaccional** con la base de datos que debemos abrir y mantener abierta mientras estamos realizado operaciones. En este sentido, realiza **funciones similares** a las de una **conexión JDBC**.
  - Además, mantiene en **memoria una caché** con las entidades que gestiona y es responsable de sincronizarlas correctamente con la base de datos cuando se realiza un flush. El conjunto de entidades que gestiona un entity manager se denomina su **contexto de persistencia**.

Introducción

- El **entity manager** se obtiene a través de una factoría del tipo **EntityManagerFactory**, que se configura mediante la especificación de **una unidad de persistencia** (persistence unit en inglés) definida en el **fichero XML persistence.xml**.
- En el fichero pueden haber definidas más de una unidad de persistencia, cada una con un nombre distinto. El nombre de la unidad de persistencia escogida se pasa a la factoría.
- La unidad de persistencia define las **características concretas de la base de datos** con la que van a trabajar todos los entity managers obtenidos a partir de esa factoría y queda asociada a ella en el momento de su creación.
- Existe, por tanto, una **relación uno-a-uno** entre una unidad de persistencia y su **EntityManagerFactory** concreto.
- Para obtener una **factoría EntityManagerFactory** debemos llamar a un **método estático de la clase Persistence**.

Introducción

- Las **relaciones** entre las clases que intervienen en la **configuración** y en la creación de entity managers se muestran en la **siguiente figura**:



Introducción

- Una vez creado el **entity manager** lo utilizaremos para realizar todas las operaciones de recuperación, consulta y actualización de entidades.

- Cuando un entity manager obtiene **una referencia a una entidad**, se dice que la entidad está **gestionada** (**managed entity** en inglés) por él.

- El **entity manager** guarda internamente **todas las entidades** que gestiona y las utiliza como una **caché de los datos** en la base de datos. Por ejemplo, cuando va a recuperar una **entidad por su clave** primaria, lo **primero** que hace es **consultar en su caché** si esta entidad ya la ha recuperado previamente. Si es así, **no necesita hacer la búsqueda en la base de datos** y devuelve la propia referencia que mantiene. Al conjunto de entidades gestionadas por un entity manager se le denomina su **contexto de persistencia** (persistence context).

- En un determinado momento, el **entity manager** debe **volcar** a la base de datos todos los **cambios** que se han realizado sobre las entidades. También debe ejecutar las **consultas JPQL definidas**. Para ello el entity manager utiliza **un proveedor de persistencia** (persistence provider en inglés) que es el responsable de generar todo el **código SQL compatible con la base de datos**.

El fichero de persistencia

- Introducción.
- Esquema xml.
- Elementos fundamentales.
- Propiedades.

Introducción

- En primer lugar, vamos a hablar del **fichero persistence.xml** que se encuentra ubicado en la **carpeta META-INF** del proyecto.
- Un archivo **persistence.xml** define una **unidad de persistencia**.
- Este fichero se encarga de **conectarnos a la base de datos** y define el **conjunto de entidades** que vamos a gestionar.
- El fichero **es parte del estándar** y existirá en cualquier implementación de JPA que se utilice.

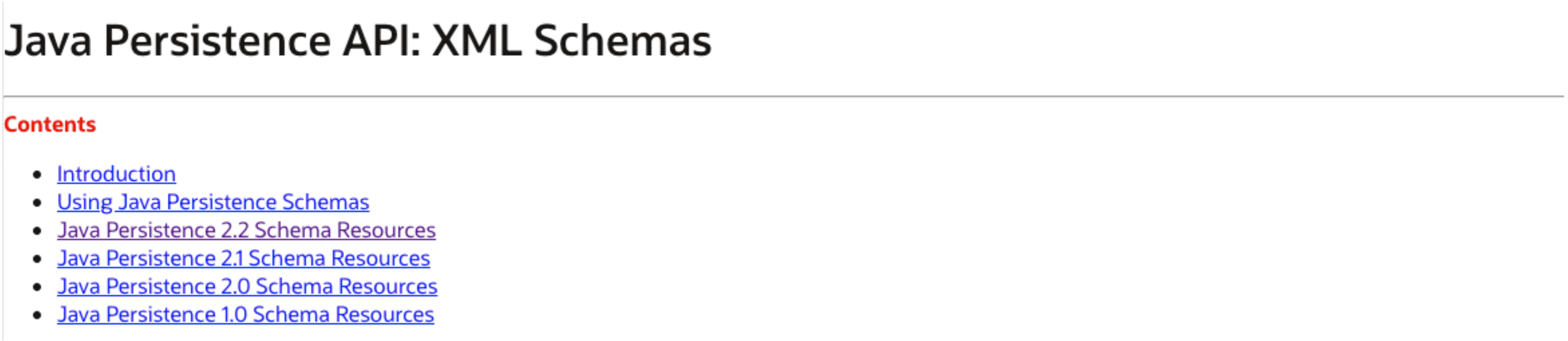
Introducción

- Ejemplo:

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd" version="2.0">
 <persistence-unit name="UnidadPersonas">
 <class>es.curso.bo.Persona</class>
 <properties>
   <property name= "hibernate.show_sql" value="true" />
   <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
   <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
   <property name="javax.persistence.jdbc.user" value="root" />
   <property name="javax.persistence.jdbc.password" value="jboss" />
   <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/jpa" />
</properties>
</persistence-unit>
```

Esquema xml

- Los esquemas xml que definen este fichero se encuentran en:
  - http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/persistence/index.html



Java Persistence API: XML Schemas

Contents

- Introduction
- Using Java Persistence Schemas
- Java Persistence 2.2 Schema Resources
- Java Persistence 2.1 Schema Resources
- Java Persistence 2.0 Schema Resources
- Java Persistence 1.0 Schema Resources

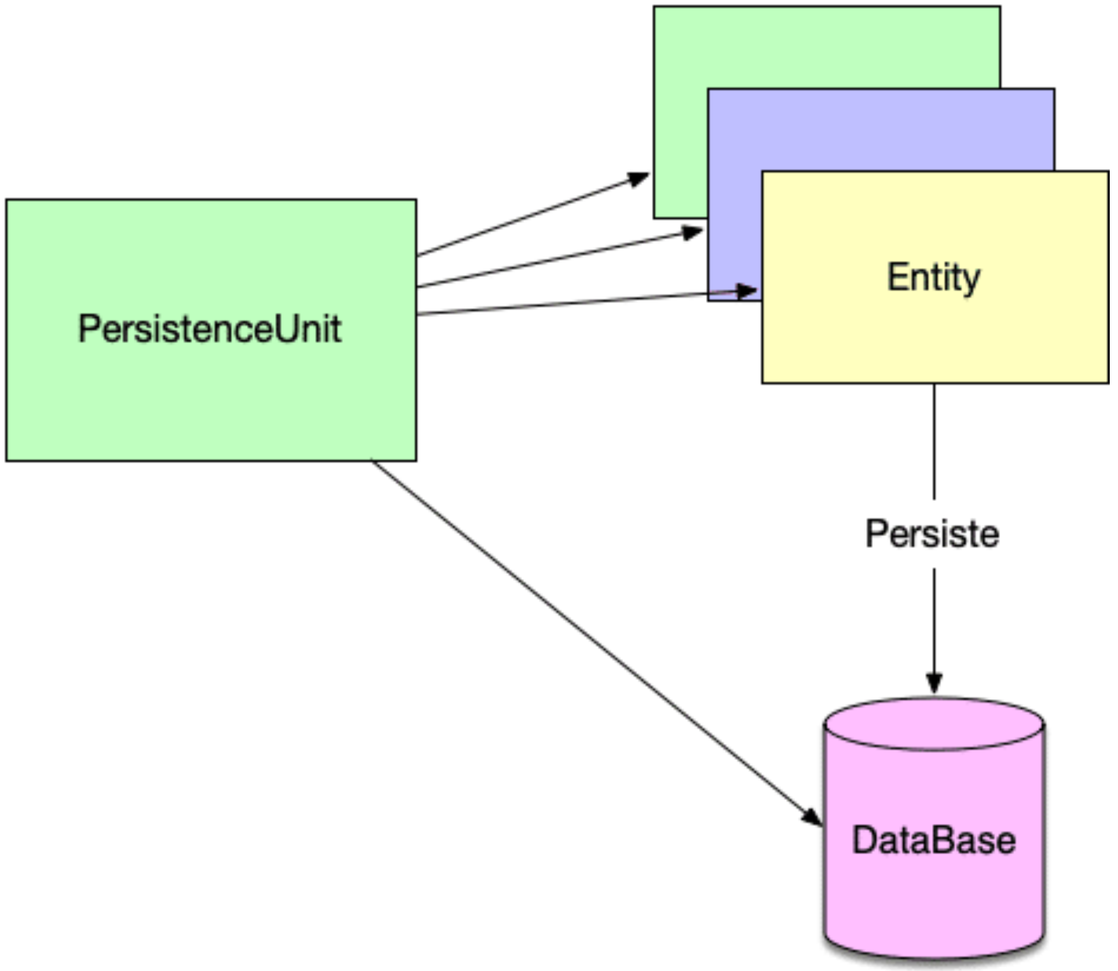
Esquema xml

- El esquema de la versión 2.2 se encuentra en la siguiente url:
  - http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/persistence/persistence_2_2.xsd
- Por lo tanto, el fichero persistence.xml para la versión 2.2 tendrá la siguiente apariencia:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
         http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/persistence/persistence_2_2.xsd"
        version="2.2">
</persistence>
```

Elementos fundamentales

- En el documento XML podemos ver tanto **nuestras clases de persistencia** como las **propiedades de conexión a la base de datos**.



Elementos fundamentales

- Ejemplo: **unidad de persistencia**:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
        http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/persistence/persistence_2_2.xsd"
        version="2.2">
  <persistence-unit name="OrderManagement" />
</persistence>
```

- Se crea una unidad de persistencia denominada **OrderManagement**.
- Cualquier **clase de persistencia administrada anotada** que se encuentre en la raíz de la unidad de persistencia se agrega a la lista de clases de persistencia administrada.

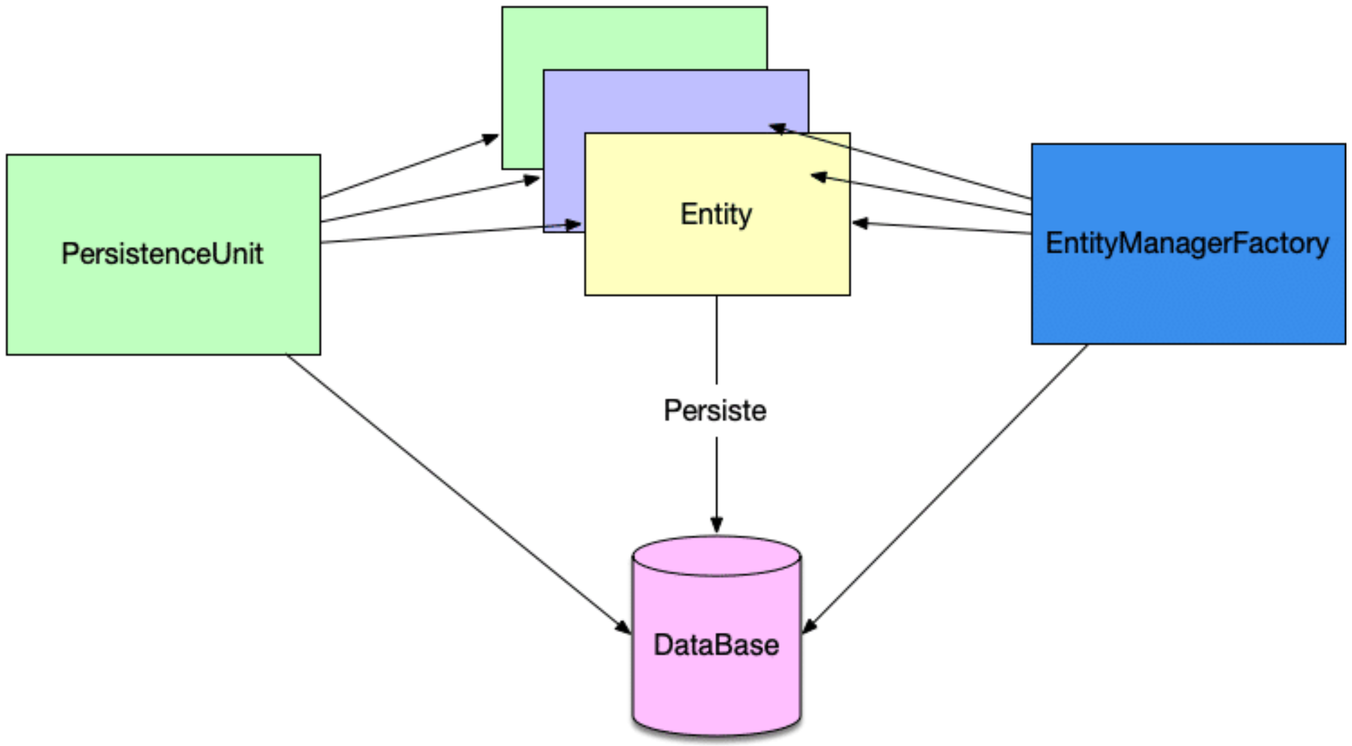
Elementos fundamentales

- Ejemplo: **unidad de persistencia**:
  - Si existe un **archivo META-INF/orm.xml**, las clases a las que hace referencia y la información de asignación que contiene se utilizan como se especifica anteriormente.
  - Debido a que no se especifica ningún proveedor, se supone que la unidad de persistencia es portátil entre proveedores.
  - Debido a que no se especifica el tipo de transacción, se asume JTA para entornos Java EE. El contenedor debe proporcionar la fuente de datos (se puede especificar en la implementación de la aplicación, por ejemplo).
  - En **entornos Java SE**, la fuente de datos puede especificarse por otros medios y se asume un tipo de transacción de **RESOURCE_LOCAL**.

Elementos fundamentales

- Debemos tener en cuenta que en **Java Persistence API** el **fichero persistence.xml** define la conectividad a la base de datos y las entidades que vamos a usar.

- Como resultado de ambos conceptos se genera un objeto **EntityManagerFactory** que se encargará de gestionar todas estas entidades para la base de datos.



Elementos fundamentales

- Las entidades pueden definirse de diferentes formas, por ejemplo, incluyendo el nombre de las clases gracias al **elemento class**:

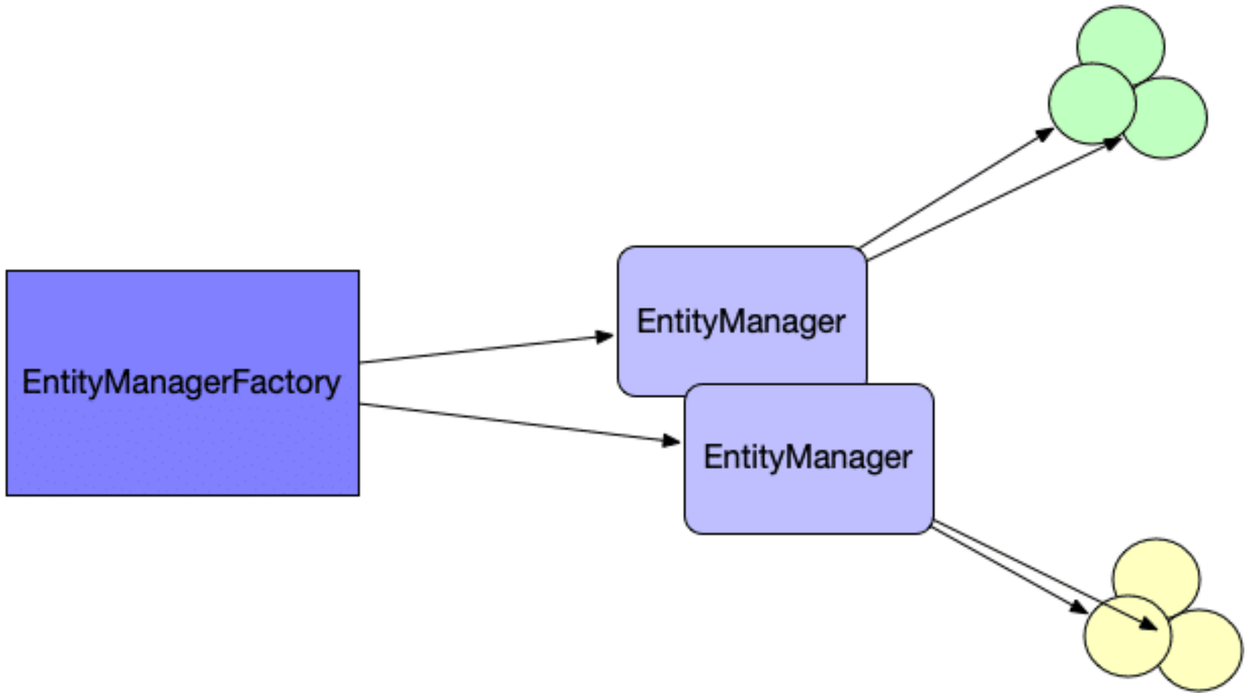
```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
       http://www.oracle.com/webfolder/technetwork/jsc/xml/ns/persistence/persistence_2_2.xsd"
        version="2.2">

        <persistence-unit name="EmployeePU">
                <provider>org.hibernate.ejb.HibernatePersistence</provider>
                <class>com.curso.jpa.entity.Employee</class>
                ...
        </persistence-unit>
</persistence>
```

Elementos fundamentales

- De esta forma tendremos a nuestra disposición un **EntityManagerFactory** con el que empezar a gestionar las entidades que se encuentran definidas a nivel del fichero persistence.xml. Eso si, muchas aplicaciones JEE conectan a varias bases de datos y tendrán diferentes EntityManagerFactorys. Cada uno estará ligado un PersistenceUnit diferente.

- Una vez disponemos de un **EntityManagerFactory** , este será capaz de construir un objeto **EntityManager** que automatizará la persistencia de los objetos.



Propiedades

- Algunas de ellas son:
  - javax.persistence.jdbc.driver.
  - javax.persistence.jdbc.url.
  - javax.persistence.jdbc.user.
  - javax.persistence.jdbc.password.
- Para más información, consultar la especificación estándar.

Propiedades

- Como vamos a trabajar con Hibernate también podremos incluir alguna de sus propiedades:
  - hibernate.dialect.
  - hibernate.show_sql.
  - hibernate.hbm2ddl.auto.

Propiedades

- Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence ...>
  <persistence-unit name="EmployeePU">
    ...
    <properties>
      <property name="hibernate.dialect" value="org.hibernate.dialect.MySQL5Dialect"/>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="javax.persistence.jdbc.driver" value="com.mysql.cj.jdbc.Driver"/>
      <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost/jpa?serverTimezone=UTC"/>
      <property name="javax.persistence.jdbc.user" value="jpa"/>
      <property name="javax.persistence.jdbc.password" value="jpa"/>
    </properties>
  </persistence-unit>
</persistence>
```

Obtención del EntityManager

- La forma de **obtener un entity manager** varía dependiendo de si estamos utilizando JPA gestionado por la aplicación (cuando **utilizamos JPA en Java SE**) o si estamos utilizando JPA en una aplicación gestionada por **un servidor de aplicaciones Java EE**.

- En el **segundo caso** se utiliza un método denominado inyección de dependencias y el servidor de aplicaciones será el responsable de obtener el entity manager e inyectarlo en una variable que tiene una determinada anotación. Lo veremos más adelante.

- En el **primer caso**, cuando estamos usando JPA gestionado por la aplicación, el entity manager se obtiene a partir de un **EntityManagerFactory**. Para obtener la factoría se debe llamar al método estático **createEntityManagerFactory()** de la clase Persistence. En este método se debe proporcionar el nombre de la unidad de persistencia que vamos a asociar a la factoría.

- Por ejemplo, para obtener un EntityManagerFactory asociado a la unidad de persistencia llamada "simplejpa" hay que escribir lo siguiente:

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("simplejpa");
```

Obtención del EntityManager

- El nombre "**simplejpa**" indica el nombre de la unidad de persistencia en la que se especifican los parámetros de configuración de la conexión con la base de datos (URL de la conexión, nombre de la base de datos, usuario, contraseña, gestor de base de datos, características del pool de conexiones, etc.). Esta unidad de persistencia se especifica en el fichero estándar de JPA META-INF/ persistence.xml.

- Una vez que tenemos una factoría, podemos obtener fácilmente un EntityManager:

```
EntityManager em = emf.createEntityManager();
```

- Esta **llamada no es demasiado costosa**, ya que las implementaciones de JPA (como Hibernate) implementan **pools de entity managers**.

- El método **createEntityManager** no realiza ninguna reserva de memoria ni de otros recursos sino que, simplemente, devuelve alguno de los entity managers disponibles.

Obtención del EntityManager

- Ejemplo:

```
public class AutorTest {
  public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("simplejpa");
    EntityManager em = emf.createEntityManager();
    EntityTransaction tx = em.getTransaction();
    tx.begin();
    Autor autor = em.find(Autor.class, "dennis.ritchie@gmail.com");
    Mensaje mensaje = new Mensaje("Hola mundo", autor);
    em.persist(mensaje);
    autor.getMensajes().add(mensaje);

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

Obtención del EntityManager

- Primero, se obtiene el **entity manager** a partir de la llamada al método **createEntityManager** de la clase Persistence.
- Después, se marca el **comienzo de una transacción** y se recupera un autor de la base de datos con la **llamada a find**.
- Después se **crea un nuevo mensaje**, que se incorpora al contexto de persistencia con el método persist.
- Por último se actualiza la colección con los mensajes creados por el autor y se cierra la transacción y la entidad de persistencia.

Obtención del EntityManager

- Es muy importante considerar que **los objetos EntityManager no son thread-safe**.
- Cuando los utilicemos en **servlets**, por ejemplo, deberemos crearlos **en cada petición HTTP**.
- De esta forma se evita que distintas sesiones accedan al mismo contexto de persistencia. Si queremos que una sesión HTTP utilice un único entity manager, podríamos guardarlo en el objeto HtttpSession y acceder a él al comienzo de cada petición. El objeto EntityManagerFactory a partir del que obtenemos los entity managers sí que es thread-safe.

Obtención del EntityManager

- Podemos implementar un **singleton** al que acceder para **obtener entity managers**. Lo llamamos **PersistenceManager** y lo definimos en el paquete persistence en el que vamos a crear la capa de persistencia:

```
...
public class PersistenceManager {
  static private final String PERSISTENCE_UNIT_NAME = "simplejpa";
  protected static PersistenceManager me = null;
  private EntityManagerFactory emf = null;

  private PersistenceManager() {
    if (emf == null) {
      emf = Persistence.createEntityManagerFactory(PERSISTENCE_UNIT_NAME);
      this.setEntityManagerFactory(emf);
    }
  }
```

Obtención del EntityManager

- (cont):

```
public static PersistenceManager getInstance() {
  if (me == null) {
    me = new PersistenceManager();
  }
  return me;
}

public void setEntityManagerFactory(EntityManagerFactory emf) {
  this.emf = emf;
}
```

Obtención del EntityManager

- (cont.):

```
  public EntityManagerFactory getEntityManagerFactory() {
    return this.emf;
  }

  public EntityManager createEntityManager() {
    return emf.createEntityManager();
  }
}
```

- En la clase se define una cadena estática con el nombre de la unidad de persistencia que carga el singleton, en este caso **simplejpa**.

Obtención del EntityManager

- **Para obtener el entity manager** debemos hacer ahora lo siguiente:

```
public class AutorTest {
  public static void main(String[] args) {
    EntityManager em = PersistenceManager.getInstance().createEntityManager();
    em.getTransaction().begin();

    Autor autor = em.find(Autor.class, "dennis.ritchie@gmail.com");
    Mensaje mensaje = new Mensaje("Hola mundo", autor);
    em.persist(mensaje);
    autor.getMensajes().add(mensaje);

    em.getTransaction().commit();
    em.close();
  }
}
```

Contexto de persistencia

- Una cuestión muy importante para entender el **funcionamiento** del entity manager es comprender su **contexto de persistencia**. Contiene la **colección de entidades gestionadas por el entity manager** que están conectadas y sincronizadas con la base de datos.

- Cuando el entity manager **cierra una transacción**, su contexto de persistencia se **sincroniza** automáticamente con la base de datos.

- Sin embargo, **a pesar del importante papel que juega**, el contexto de persistencia **nunca es** realmente **visible** a la aplicación. Siempre se accede a él indirectamente a través del entity manager y asumimos que está ahí cuando lo necesitamos.

- Es también **fundamental** entender que el contexto de persistencia hace el papel de **caché** de las entidades que están realmente en la base de datos. Cuando actualizamos una instancia en el contexto de persistencia estamos **actualizando una caché**, una copia que sólo se hace persistente en la base de datos cuando el **entity manager realiza un flush** de las instancias en la base de datos.

Contexto de persistencia

- Simplificando bastante, podemos pensar que el **entity manager realiza el siguiente proceso** para todas las entidades:
  - Si la aplicación **solicita una entidad** (mediante un find, o accediendo a un atributo de otra entidad en una relación), se comprueba **si ya se encuentra en el contexto de persistencia**. Si es así, se devuelve su referencia. Si no se ha recuperado previamente, se obtiene la instancia de la entidad de la base de datos.
  - La aplicación utiliza las entidades del contexto de persistencia, accediendo a sus atributos y (posiblemente) **modificándolos**. Todas las **modificaciones se realizan en la memoria**, en el contexto de persistencia.
  - En un momento dado (cuando termina la **transacción**, se ejecuta una **query** o se hace una llamada al **método flush**) el entity manager **comprueba qué entidades han sido modificadas** y vuelca los cambios a la **base de datos**.

Contexto de persistencia

- Es muy importante darse cuenta de la **diferencia** entre el **contexto** de persistencia y la **base de datos** propiamente dicha.
- La **sincronización** no se realiza hasta que el entity manager vuelca los cambios a la base de datos.
- La **aplicación debe ser consciente de esto** y utilizar razonablemente los contextos de persistencia.