Java Persistence API

- Sesión 2: Conceptos básicos de JPA

Índice

- Entidades
- Características de una entidad
- Entity Manager
- DAOs en JPA
- EAO: Entity Access Object

Entidades persistentes

- Una entidad agrupa un conjunto de datos de forma coherente. Se corresponde con una tabla de una base de datos. Las instancias de la entidad se corresponden con filas de la tabla.
- Ejemplos: Hotel, Vuelo, Estudiante, ...
- ¿Son las entidades *objetos persistentes*? No: un objeto persistente se hace persistente automáticamente cuando se crea (cuando se hace un `new()`) y los valores de sus atributos están *directamente* ligados a la base de datos.
- Una entidad JPA es un POJO, un Puro Viejo Objeto Java, con unas anotaciones. La aplicación es la responsable de manejar la persistencia con las llamadas (implícitas o explícitas) al entity manager y a las transacciones.

Características de las entidades (1)

- A pesar de ser POJOs, las entidades deben cumplir un conjunto de condiciones para poder ser persistentes
- Características:
  - *Atributos*. Tienen un conjunto de atributos que son tipos Java u otras entidades.
  - *Identidad*. Uno de los atributos de la entidad debe ser el encargado de identificar de forma única sus instancias de la entidad. Corresponderá con la clave primaria de la tabla en la que se mapea la entidad.
  - *Transaccionalidad*. La sincronización de las entidades en memoria a la base de datos es transaccional (hay que definir un comienzo y un `commit()` y si hay algún fallo se realiza un `rollback()` y se deshacen todos los cambios). En memoria, sin embargo, no existe esta transaccionalidad.

Características de las entidades (2)

- Más características:
  - *Granularidad*. Las entidades deben tener una granularidad justa. Ni demasiado pequeña (un string) ni demasiado grande (100 columnas). Lo normal es definir entidades ligeras.
  - *Metadatos*. Las características de persistencia de las entidades se definen con metadatos en forma de anotaciones o ficheros XML. JPA consulta estos metadatos para gestionar la persistencia de la entidad.

Un ejemplo de entidad

```
@Entity
public class Empleado {
   @Id
   private int id;
   private String nombre;
   private long salario;

   public Empleado {}
   public Empleado(int id) { this.id = id; }

   public int getId() { return id; }
   public void setId(int id) { this.id = id; }
   public String getNombre() { return nombre; }
   public void setNombre(String nombre) { this.nombre = nombre; }
   public long getSalario() { return salario; }
   public void setSalario(long salario) { this.salario = salario; }
}
```

EntityManagerFactory (1)

- Normalmente se crea una única vez
- Se le pasa como parámetro el nombre de una unidad de persistencia definida en el fichero `META-INF/persistence.xml`
- En la unidad de persistencia se define la configuración de la base de datos a la que va a acceder el entity manager: nombre, driver, contraseñas, ...
- El entity manager se obtiene con un método de la clase EntityManagerFactory

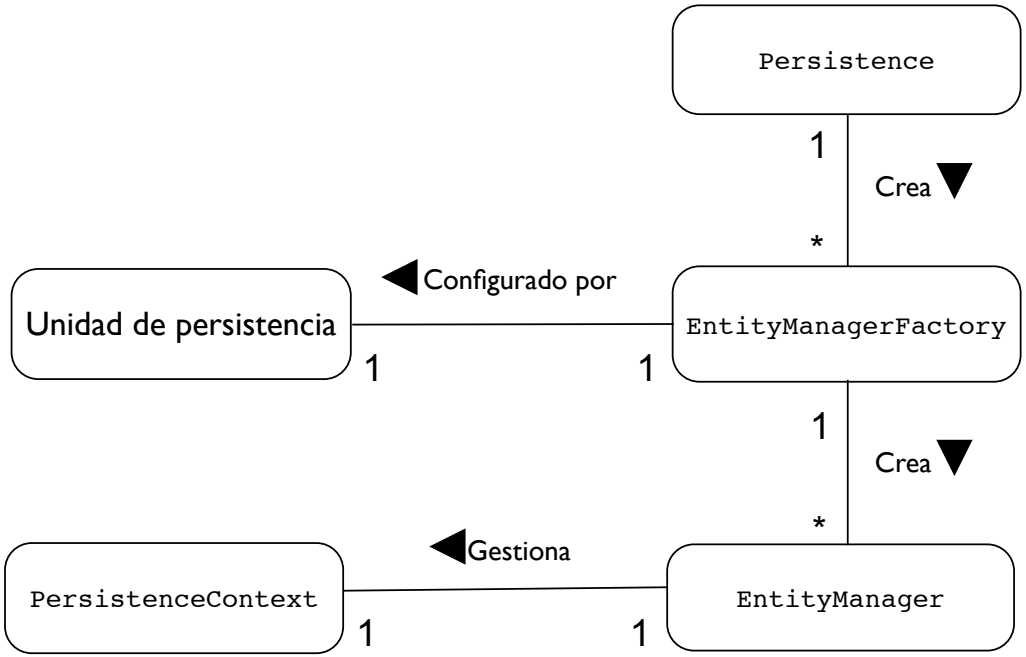
```
EntityManagerFactory emf =
   Persistence.createEntityManagerFactory("simplejpa");
EntityManager em = emf.createEntityManager();
```

EntityManagerFactory (2)

- Ejemplo de fichero `META-INF/persistence.xml`

```
<persistence-unit name="simplejpa">
   <properties>
      <property name="hibernate.connection.driver_class"
                value="com.mysql.jdbc.Driver" />
      <property name="hibernate.connection.url"
                value="jdbc:mysql://localhost:3306/jpa" />
      <property name="hibernate.connection.username"
                value="root" />
      <property name="hibernate.connection.password"
                value="root" />
      <property name="hibernate.dialect"
                value="org.hibernate.dialect.MySQL5Dialect" />
      <property name="hibernate.hbm2ddl.auto"
                value="update" />
      <property name="hibernate.show_sql"
                     value="true"/>
      ...
   </properties>
```

Relaciones del entity manager



Operaciones del entity manager

- El entity manager es el responsable de mantener las entidades sincronizadas con la base de datos
- Operaciones sobre las entidades:
  - Hacer persistente una entidad
  - Buscar una entidad en la BD
  - Borrar entidades
  - Actualizar entidades
  - Queries a la BD

Creando entidades

- Una vez creada una instancia `entity` como un objeto Java normal, hay que llamar al método `persist(entity)` para hacerla persistente
- La instancia pasará a formar parte del contexto de persistencia y será gestionada por el entity manager
- El entity manager generará una sentencia SQL `INSERT` la siguiente vez que se realice un flush
- Si la entidad tiene un identificador ya existente en la base de datos se generará una excepción runtime `javax.persistence.EntityExistsException`

```
public Empleado createEmpleado(int id, String nombre, long salario) {
   Empleado emp = new Empleado(id);
   emp.setNombre(nombre);
   emp.setSalario(salario);
   em.persist(emp);
   return emp;
}
```

Buscando entidades

- Para buscar una entidad en la base de datos hay que llamar al método `find(clase, id)`, pasando como parámetro la clase de entidad que se busca y el identificador de la instancia
- Si no existe ninguna entidad con ese identificador, se devuelve `null`

```
public Empleado findEmpleado(int id) {
   return em.find(Employee.class, id);
}
```

Borrando entidades

- El entity manager puede borrar una entidad gestionada con el método `remove()`
- Se genera una sentencia `DELETE` la siguiente vez que se hace un *flush*

```
public void removeEmpleado(int id) {
    Empleado emp = em.find(Empleado.class, id);
    if (emp != null) {
        em.remove(emp);
    }
}
```

Modificando entidades

- Una vez que el entity manager gestiona una entidad, para modificar uno de sus atributos no hay más que llamar a un método `set` de la entidad
- La siguiente vez que se haga un *flush* se sincroniza el estado de la entidad con la BD con una sentencia `UPDATE`

```
public Empleado subeSueldoEmpleado(int id, long aumento) {
   Empleado emp = em.find(Empleado.class, id);
   if (emp != null) {
      emp.setSueldo(emp.getSueldo + aumento);
   }
}
```

Queries a la BD

- También es posible realizar consultas elaboradas a la base de datos utilizando el lenguaje JPQL
- Se devuelve un objeto o una colección a los que hay que hacer un casting (o `null` si no hay resultados)

```
public List<Empleado> findEmpleadosSueldo(long sueldo) {
   Query query = em.createQuery("SELECT e FROM Empleado e " +
                                "WHERE e.sueldo > :sueldo");
   query.setParameter("sueldo", 20000);
   return (List<Empleado>) query.getResultList();
}
```

Flush

- Es posible obligar al entity manager a sincronizar el contexto de persistencia con la base de datos usando el método `flush()`
- Otro método relacionado es `setFlushMode(FlushModeType)`, con el que se le puede indicar al entity manager cuándo hacer la sincronización
- Los posibles valores de `FlushModeType` son:
  - `FlushModeType.AUTO` (valor por defecto): se sincroniza cuando se hace un commit y previamente a una consulta
  - `FlushModeType.COMMIT`: se sincroniza cuando la transacción hace un commit

Transacciones

- Todas las operaciones de actualización del entity manager hay que hacerlas dentro de una transacción, en caso contrario se genera una excepción runtime de tipo `javax.persistence.TransactionRequiredException`
- Se obtiene una transacción con el método `beginTransaction()` del entity manager
- Dada una transacción es posible:
  - Cerrarla con `commit()`
  - Deshacerla con `rollback()`

Data Access Object

- Es muy sencillo implementar un DAO con JPA
- Un DAO recibe y devuelve objetos POJOs no conectados con la base de datos (valores, no entidades)
- Cada llamada a un método del DAO es atómica
- La separación entre la capa de aplicación y la de datos es total
- Inconveniente: perdemos toda la flexibilidad de JPA en la capa de aplicación

Código fuente (1)

```
public class EmpleadoDAO {

   protected EntityManagerFactory emf;

   public EmpleadoDAO() {
      emf = Persistence.createEntityManagerFactory("simplejpa");
   }

   public Empleado createEmpleado(int id, String nombre, long sueldo)  {
      EntityManager em = emf.createEntityManager();
      em.getTransaction().begin();
      Empleado emp = em.find(Empleado.class, id);
      if (emp == null) {
         emp = new Empleado(id);
      }
      emp.setNombre(nombre);
      emp.setSueldo(sueldo);
      em.persist(emp);
      em.getTransaction().commit();
      em.close();
      return emp;
   }

   //...
```

Código fuente (2)

```
   //...

   public void subeSueldoEmpleado(int id, long aumento) {
      EntityManager em = emf.createEntityManager();
      em.getTransaction().begin();
      Empleado emp = em.find(Empleado.class, id);
      if (emp != null) {
         emp.setSueldo(emp.getSueldo() + aumento);
      }
      em.getTransaction().commit();
      em.close();
   }

   public List<Empleado> findEmpleadosSueldoMayorQue(long sueldo) {
      EntityManager em = emf.createEntityManager();
      em.getTransaction().begin();
      Query query = em.createQuery("SELECT e FROM Empleado e " +
                                   "WHERE e.sueldo > :sueldo");
      query.setParameter("sueldo", sueldo);
      List<Empleado> list = (List<Empleado>) query.getResultList();
      em.getTransaction().commit();
      em.close();
      return list;
   }
}
```

Entity Access Object

- Nuevo “patrón” para poder utilizar entidades persistentes conectadas a la BD
- A diferencia del DAO, hay que pasarle al constructor el entity manager. De esta forma distintas llamadas a los DAO pueden compartir un mismo contexto de persistencia y una misma transacción
- El inconveniente: no se encapsula la creación del entity manager ni el uso de transacciones, con lo que obligamos a que el que use esta capa debe entender de estos conceptos

Código fuente (1)

```
public class EmpleadoEAO {
   protected EntityManager em;

   public EmpleadoEAO(EntityManager em) {
      this.em = em;
   }

   public Empleado createEmpleado(int id, String nombre, long sueldo) {
      Empleado emp = new Empleado(id);
      emp.setNombre(nombre);
      emp.setSueldo(sueldo);
      em.persist(emp);
      return emp;
   }

   public void removeEmpleado(Empleado emp) {
      em.remove(emp);
   }

   public void subeSueldoEmpleado(Empleado emp, long aumento) {
      emp.setSueldo(emp.getSueldo() + aumento);
   }

   //...
```

Código fuente (2)

```
   //...

   public void cambiaNombreEmpleado(Empleado emp, String nuevoNombre) {
      emp.setNombre(nuevoNombre);
   }
   public Empleado findEmpleado(int id) {
      return em.find(Empleado.class, id);
   }

   public List<Empleado> findEmpleadosSueldo(long sueldo) {
      Query query = em.createQuery("SELECT e FROM Empleado e "
            + "WHERE e.sueldo > :sueldo");
      query.setParameter("sueldo", 20000);
      return (List<Empleado>) query.getResultList();
   }
}
```

¿Preguntas?