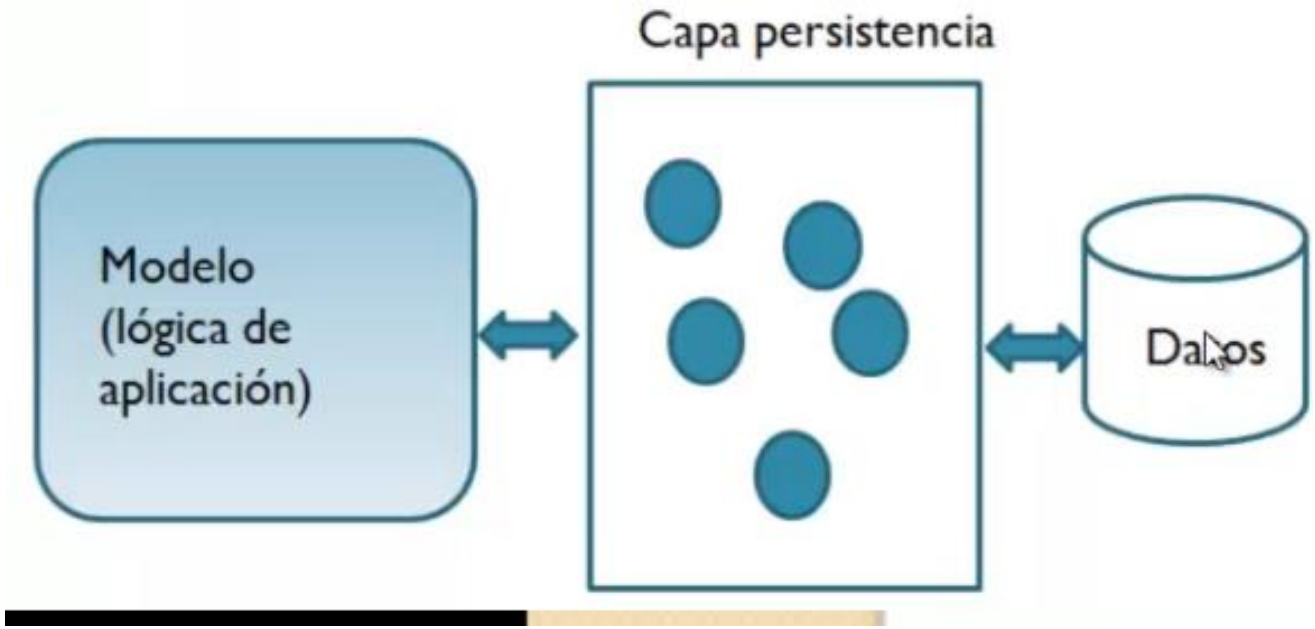
UT. 3 Persistencia con JPA

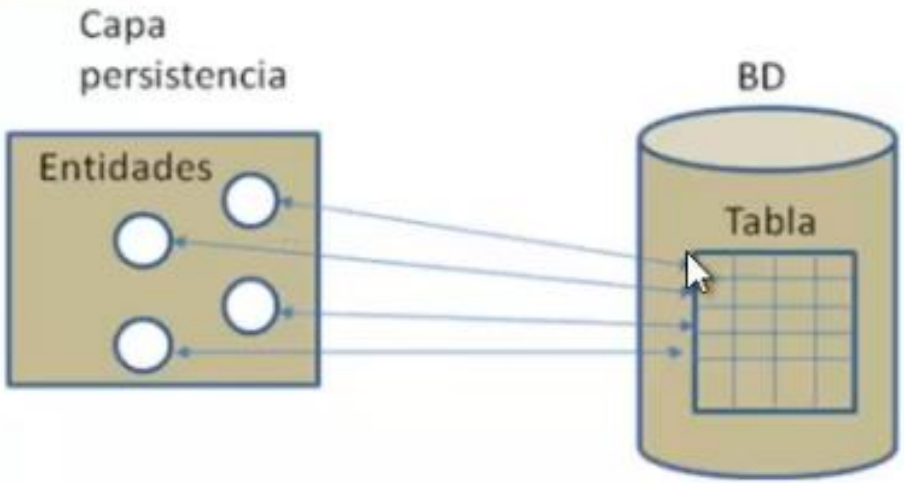
¿Qué es una capa de persistencia?

- Los datos son expuestos a la aplicación en forma de objetos (entidades).
- La lógica de negocio maneja objetos, no tablas



Entidades

- Objetos que representan una fila de una tabla de la base de datos.



- Se definen mediante clases tipo JavaBean

- Las operaciones sobre una capa de persistencia consisten en crear, modificar, eliminar y recuperar entidades.

Frameworks de persistencia

- Conjunto de utilidades que facilitan la creación y manipulación de una capa de persistencia.
- Entre los más populares:
  - Hibernate
  - Ibatis
  - JPA (no exactamente un framework)
- Componentes de un framework de persistencia
  - Motor de presistencia (mapeo ORM)
  - Sistema de configuración
  - API (acceso a la capa de persistencia)

Java Persistence API

- Especificación Java EE para la configuración y gestión de una capa de persistencia
- Universal: compatible con diferentes motores(hibernate, ibatis,..)
- API estandarizado para acceso a entidades desde una aplicación

Componentes JPA

- **Archivo de configuración persistence.xml:**
  - **Define unidades de persistencia**
  - **Cada unidad de persistencia indica el motor utilizado, propiedades de conexión a BD y lista de entidades**

```
<persistence ..>
        <persistence-unit name="unidad_1" >
                <provider>...</provider>
                <properties>
                        :
                </properties>
        </persistence-unit>
</persistence>
```

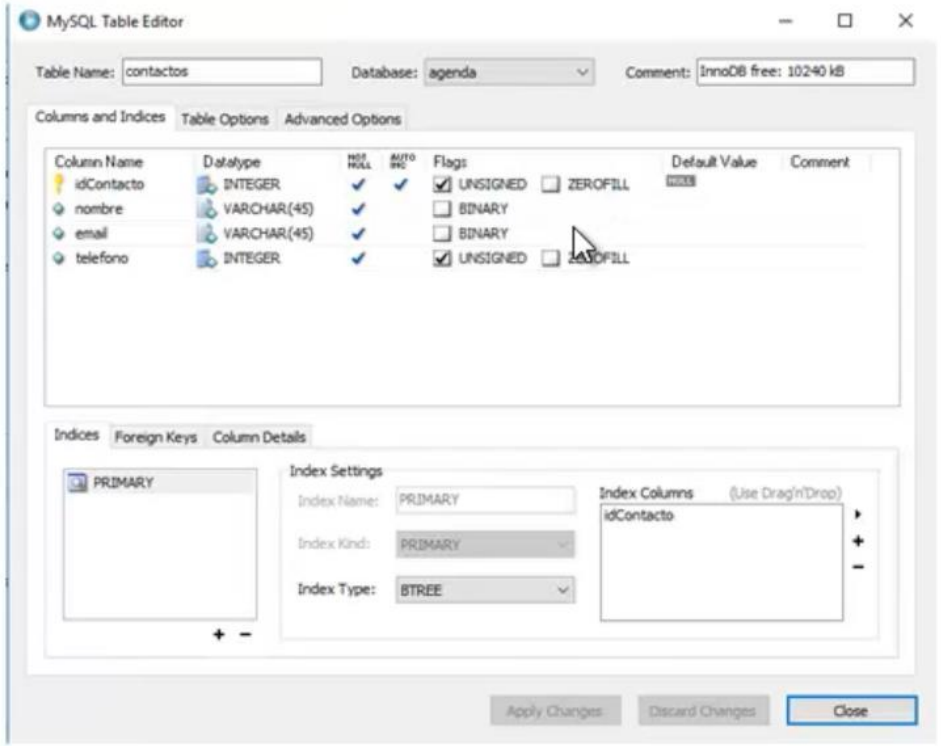
- **Conjunto de anotaciones para la configuración de entidades (@Entity, @Table, @Id,…)**
- **API JPA. Conjunto de clases e interfaces para acceso a la capa de persistencia**

Pasos para la creación de una capa de persistencia

- Creación de entidades y configuración a través de anotaciones
- Configuración de la unidad de persistencia a través del archivo persistence.xml:
  - Proveedor de persistencia
  - Lista de entidades
  - Propiedades de conexión a base de datos

Ejemplo. Crear BD

- **Creación de base de datos de ejemplo en MySQL:**
  - **Base de datos agenda**
  - **Tabla de contactos**



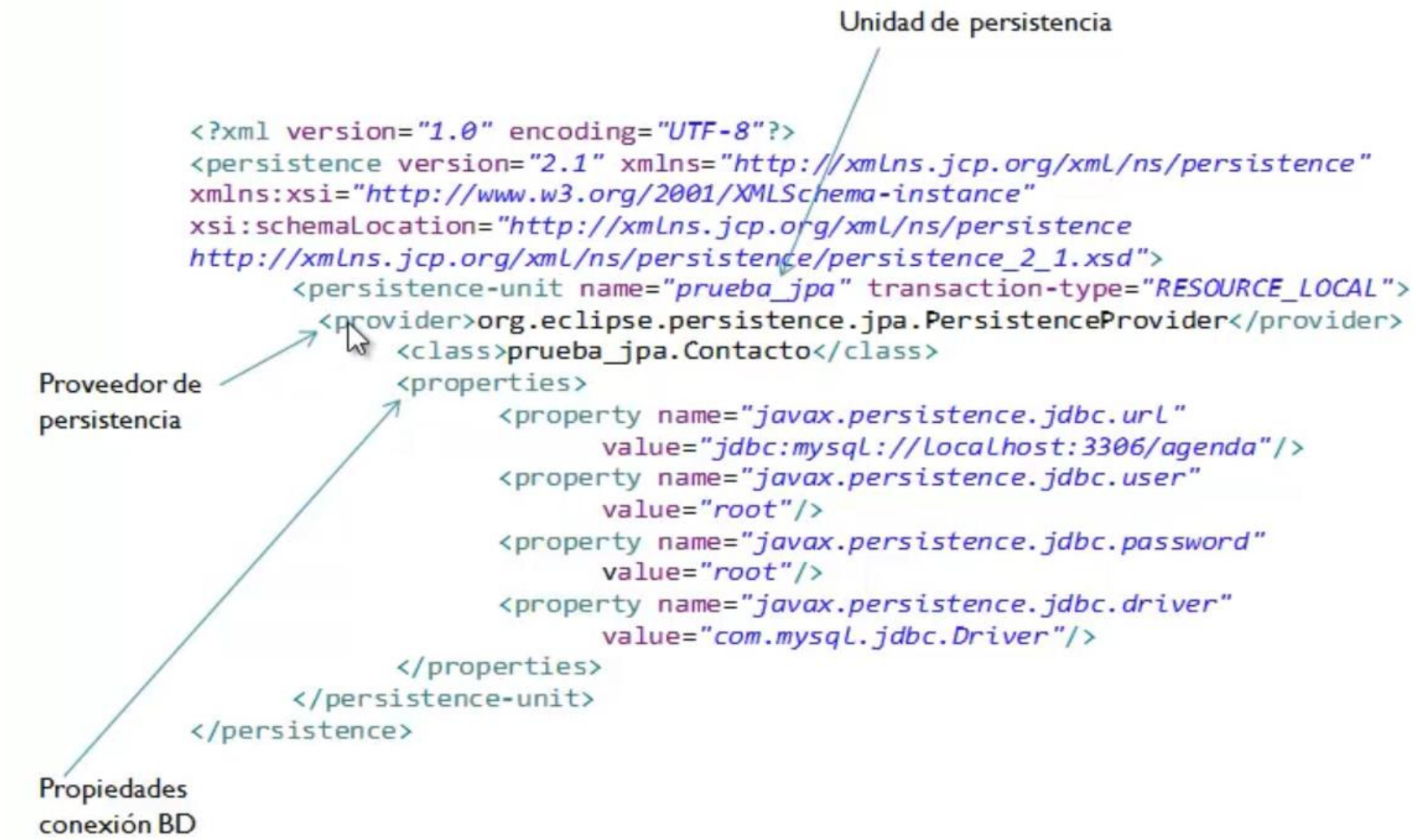
Ejemplo (cont). Creación de la Entidad contacto

```
@Entity
@Table(name="contactos")
public class Contacto implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int idContacto;
	private String email;
	private String nombre;
	private int telefono;
	public Contacto() {
	}
	public int getIdContacto() {
		return this.idContacto;
	}
	public void setIdContacto(int idContacto) {
		this.idContacto = idContacto;
	}
	public String getEmail() {
		return this.email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	:
}
```

Pricipales anotaciones

- *@Entity*. Indica que la clase es una entidad
- *@Table*. Mapea la entidad a una tabla de la base de datos
- *@Id*. Indica el atributo que contiene la primary key
- *@Column*. Asocia el atributo con una columna de la tabla. No es necesario si el nombre del atributo coincide con el de la columna.
- *@GeneratedValue*. Indica la estrategia de generación de claves en columnas primary key autogeneradas

Ejemplo (cont). persistence.xml



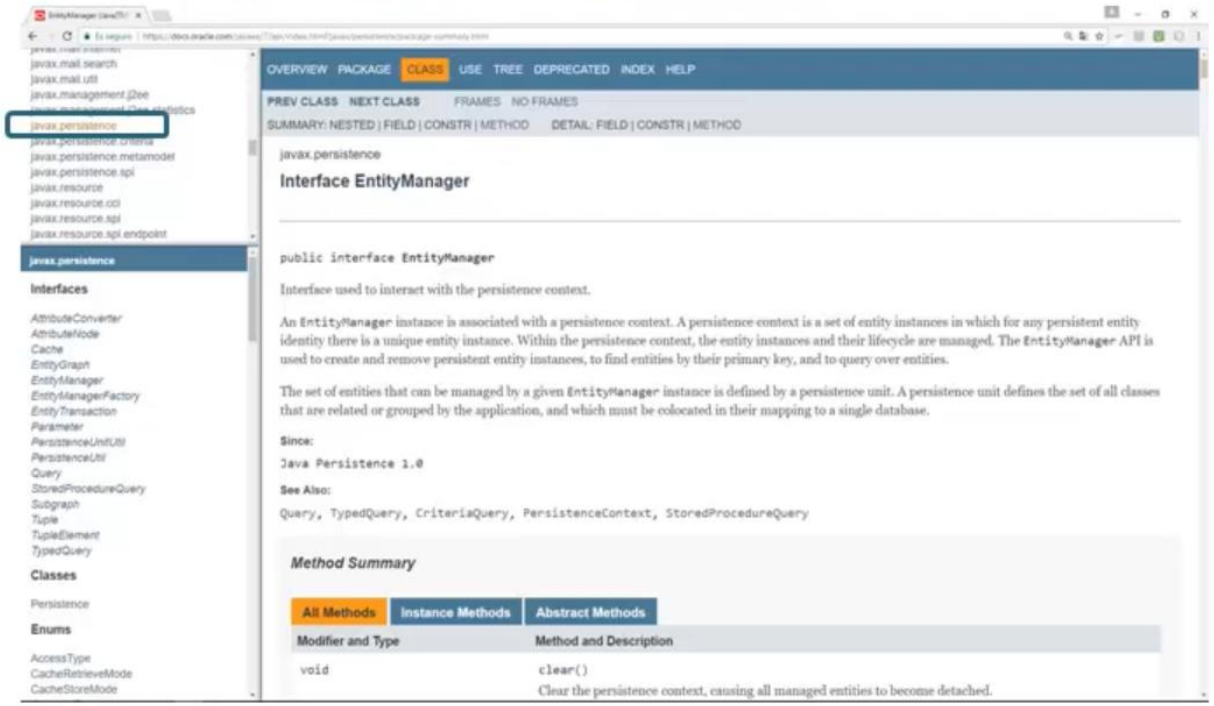

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
      <persistence-unit name="prueba_jpa" transaction-type="RESOURCE_LOCAL">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
            <class>prueba_jpa.Contacto</class>
            <properties>
                  <property name="javax.persistence.jdbc.url"
                        value="jdbc:mysql://localhost:3306/agenda"/>
                  <property name="javax.persistence.jdbc.user"
                        value="root"/>
                  <property name="javax.persistence.jdbc.password"
                        value="root"/>
                  <property name="javax.persistence.jdbc.driver"
                        value="com.mysql.jdbc.Driver"/>
            </properties>
      </persistence-unit>
</persistence>
```

El API JPA: Javadoc

- **Ayuda oficial del API:**

  https://docs.oracle.com/javaee/7/api/index.html?javax/persistence/package-summary.html

- **Paquete principal: javax.persistence**



El API JPA: Objeto EntityManager

- Implementa la interfaz javax.persistence.EntityManager
- Asociado a una unidad de persistencia
- Proporciona métodos para realizar operaciones CRUD sobre la capa de persistencia
- Se obtiene a través de un EntityManagerFactory asociado a una unidad de persistencia

El API JPA: Obtención de un EntityManager

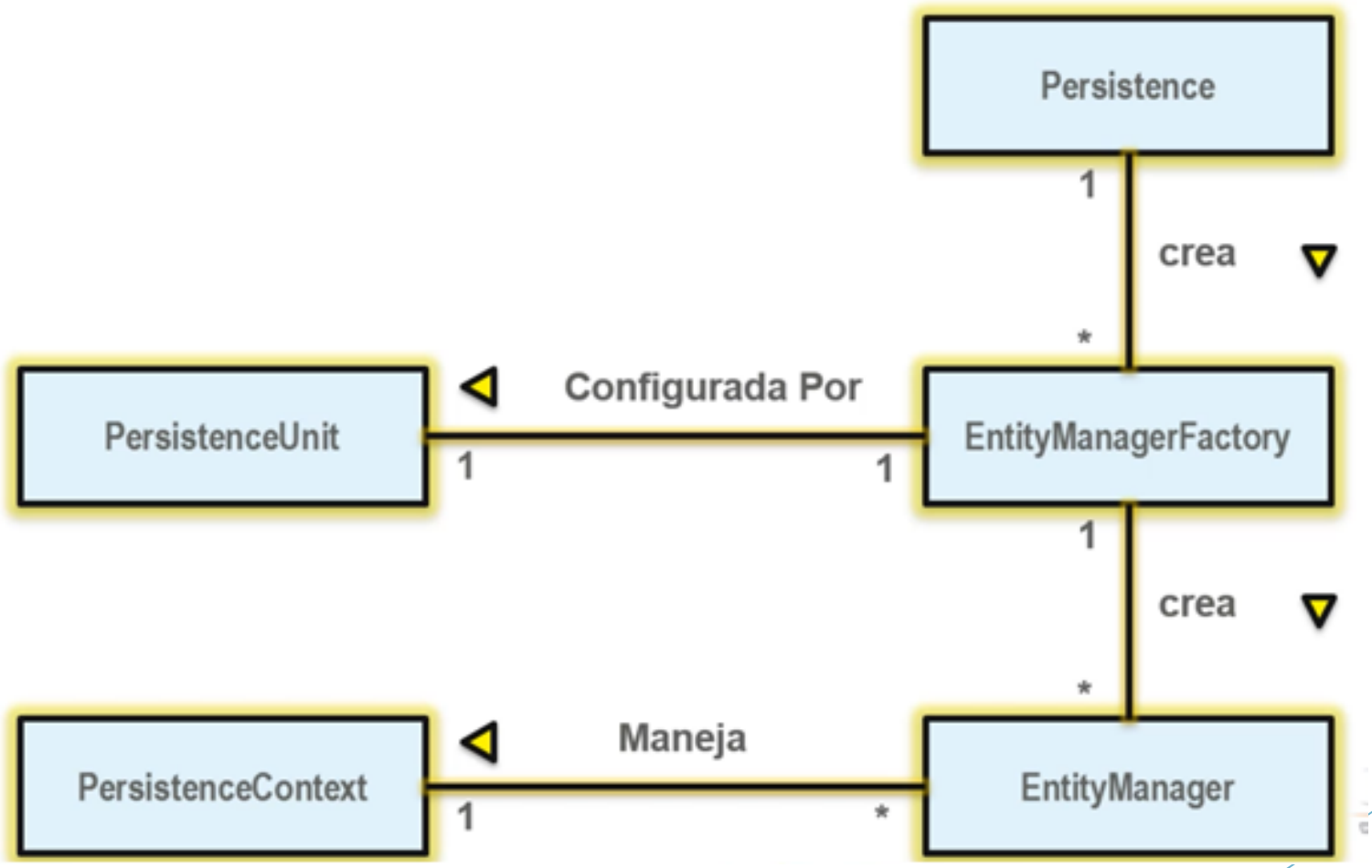
- **Proceso:**
  - **Creación de un objeto EntityManagerFactory a partir de clase Persistence.**
  - **Creación EntityManager desde EntityManagerFactory**

```
EntityManagerFactory factory;
factory=Persistence.createEntityManagerFactory("unidad_persistencia");
EntityManager em=factory.createEntityManager();
```

El API JPA: Métodos básicos de EntityManager

- **Realización de operaciones CRUD:**
  - **void persist(Object entidad). Persiste una entidad, añadiendo sus datos en una nueva fila de la BD.**
  - **void merge(Object entidad). Actualiza la BD con los datos de la entidad, que debe formar parte de la unidad de persistencia**
  - **T find(Class<T> clase_entidad, Object key). Recupera una entidad a partir de su clave primaria**
  - **void remove (Object entidad). Elimina la entidad**
- **void refresh(Object entidad). Actualiza las propiedades de la entidad con los valores de la BD**

Esquema resumen de la API JPA



El API JPA: Transacciones

- Las operaciones de acción se deben englobar en una transacción
- Gestionadas mediante EntityTransaction , que se obtiene desde EntityManager

```
EntityTransaction tx=em.getTransaction();
```

- Métodos para gestionar la Transacción:
  - begin(). Iniciar transacción
  - rollback(). Rechazar transacción
  - commit(). Confirmar transacción