EclipseLink

🧠 ¿Qué es EclipseLink?

EclipseLink es una **implementación completa de JPA (Java Persistence API)**, y también soporta otras tecnologías como:

- JAXB (para XML ↔ objetos)
- JCA (conectores de recursos)
- SDO (Service Data Objects)

Fue desarrollado como sucesor de **TopLink** y es el proveedor JPA por defecto en **Jakarta EE (antes Java EE)**.

⚙️ Integración básica con JPA

1. Añadir dependencias (Maven)

```
<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>eclipselink</artifactId>
    <version>2.7.13</version> <!-- o la versión más reciente -->
</dependency>
```

2. Configurar `persistence.xml`

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
version="2.2">
    <persistence-unit
        name="MiUnidadPersistencia" transaction-type="RESOURCE_LOCAL"
    >
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
        <class>com.ejemplo.modelo.Usuario</class>
        <properties>       <!-- Conexión a BD -->
            <property
                name="javax.persistence.jdbc.driver"
                value="com.mysql.cj.jdbc.Driver"
            />
            <property
                name="javax.persistence.jdbc.url"
                value="jdbc:mysql://localhost:3306/mi_base"
            />
            <property name="javax.persistence.jdbc.user" value="root"/>
```

```
            <property name="javax.persistence.jdbc.password" value="1234"/>
            <property name="eclipselink.logging.level" value="FINE"/>
            <property
                name="eclipselink.ddl-generation"
                value="create-or-extend-tables"
            />
        </properties>
    </persistence-unit>
</persistence>
```

📦 Weaving (optimizaciones en tiempo de ejecución)

Para aprovechar ciertas características como lazy loading, EclipseLink usa *weaving*.

Si usas Java SE:

Debes habilitar `-javaagent` en la VM:

```
-javaagent:/ruta/a/eclipselink.jar
```

🔎 Consultas EclipseLink

1. JPQL (igual que en JPA estándar)

```
TypedQuery<Usuario> q = em.createQuery(
    "SELECT u FROM Usuario u WHERE u.nombre = :nombre", Usuario.class
    );
    q.setParameter("nombre", "Luis");
```

2. Native SQL

```
Query q = em.createNativeQuery(
    "SELECT * FROM usuarios WHERE id = ?", Usuario.class
    );
    q.setParameter(1, 5);
```

3. Criteria API

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery<Usuario> cq = cb.createQuery(Usuario.class);
Root<Usuario> root = cq.from(Usuario.class);
```

```
cq.select(root).where(cb.equal(root.get("email"), "luis@correo.com"));
List<Usuario> resultados = em.createQuery(cq).getResultList();
```

💡 Opciones útiles en EclipseLink

Propiedad	Descripción
`eclipselink.logging.level`	Nivel de log (OFF, SEVERE, WARNING, INFO, FINE, etc.)
`eclipselink.ddl-generation`	Crea/modifica tablas automáticamente
`eclipselink.cache.shared.default`	Activa/desactiva cache de segundo nivel
`eclipselink.target-database`	Optimiza para un tipo específico de BD (ej. MySQL, Oracle)
`eclipselink.weaving`	Activa weaving automático o estático

🧹 Borrar y cerrar

```
EntityManagerFactory emf =
Persistence.createEntityManagerFactory("MiUnidadPersistencia");
EntityManager em = emf.createEntityManager();
em.getTransaction().begin();
Usuario u = em.find(Usuario.class, 1L);
em.remove(u);
em.getTransaction().commit();
em.close();
emf.close();
```
