

ZOO DB

OBJECT ORIENTED DATABASE MANAGEMENT SYSTEM

Introducción a las Bases de Datos Orientadas a Objetos y [ZooDB](#)

¿QUÉ ES UNA BASE DE DATOS ORIENTADA A OBJETOS?

Es una colección de objetos durable (persistida), a la que se le agregan las propiedades ACID para trabajarla.

¿QUÉ BENEFICIOS OBTENGO AL UTILIZAR UNA BD ORIENTADA A OBJETOS?

Diseño Natural: Mi modelo de objetos no se ve afectado por cuestiones de persistencia.

Transparencia!

¿REEMPLAZARÁN A LAS BASES DE DATOS RELACIONALES?

No, de ninguna manera. ¿Por qué no?

- Las BD relacionales son un excelente producto
- Recursos Humanos Capacitados
- Madurez en el Mercado
- Gran variedad de herramientas de soporte, monitoreo y backup
- Sistemas funcionando actualmente
- Miedo al cambio

¿EXISTE MUCHA OFERTA DE BASES DE DATOS ORIENTADAS A OBJETOS?

| Versant | Gemstone | Objetivity/DB |
|---------|------------|---------------|
| Perst | VelocityDB | ObjectDB |
| Newt DB | NEO | EyeDB |
| ZooDB | NDatabase | CoreObject |

y mas en nosql-database.org

¿POR QUÉ VAMOS A USAR UNA BDOD EN ESTA MATERIA?

- Es un curso de alternativas a la persistencia relacional... :)
- Son seriamente utilizadas en la **industria y la ciencia**
- Programamos en Objetos
- Necesitamos entender y practicar sober persistencia por alcance

Speaker notes

Podremos lograr la persistencia por alcance persistiendo en una BD Relacional utilizando un ORM



BD ORIENTADA A OBJETOS

VEAMOS SU API BASADA EN JDO

¿QUÉ ES JDO?

- JDO = Java Data Objects. Una API estandar para acceder datos persistentes. Ver [Apache JDO](#).
- Las interfaces más importantes de la API son:
 - *PersistentManager*: Es el contexto de persistencia, que ya vimos de que se trata
 - *Query*: El componente encargado de construir consultas y obtener resultados. (😴 Ojo, no todo implementado)
 - *Transaction*: El componente encargado de iniciar y completar transacciones

¿QUÉ REQUIERO PARA UTILIZAR ZOODB?

- Eclipse + Java 8
- Última versión de [ZooDB](#)
- Y sus depedencias: jta, jdo-api y slf4j-api

¿CÓMO USO ZOODB?

ESTRUCTURA DE UNA TRANSACCIÓN

```
ZooHelper.createDb("dbfile.zoodb");
PersistenceManager pm = ZooJdoHelper.openDB(dbName);
pm.currentTransaction().begin();
try {
    //Usar pm aca...
    pm.currentTransaction().commit();
} catch (Exception e) {
    pm.currentTransaction().rollback();
    throw e;
} finally {
    pm.close();
    pm.getPersistenceManagerFactory().close();
}
```

Speaker notes

pm es el Contexto de Persistencia del que hablamos antes

¿CÓMO USO ZOODB?

OBJETOS PERSISTENTES

Los objetos persistentes deben extender de
org.zoodb.api.impl.ZooPC

```
public class Categoria extends ZooPC {  
    private Long id;  
    private String nombre;  
    //no-arg constructor requerido por JDO  
    private Categoria() {  
  
    }  
    ...  
}
```

¿CÓMO USO ZOOODB?

PERSISTIR UN OBJETO TRANSIENT

```
...
try {
    Categoria c = new Categoria(1L, "nombre categoria");
    pm.makePersistent(c);
}
...
```

¿CÓMO USO ZOOODB?

TRAER UNA INSTANCIA POR ID

```
...
try {
    Query query = pm.newQuery(Categoría.class, "id == 1");
    Collection<Categoría> cat = (Collection<Categoría>) query.execute();
    Categoría c = cat.stream().findFirst().get();
}
...
```

¿CÓMO USO ZOOODB?

TRAER TODAS LAS INSTANCIAS DE UNA CLASE

```
...
try {
    Extent ext = pm.getExtent(Categoría.class);
    for (Categoría p : ext) {
        System.out.println("Categoría encontrada: " + p.nombre());
    }
}
...
```

¿CÓMO USO ZOOODB?

MODIFICAR UNA INSTANCIA

```
...  
try {  
    Query query = pm.newQuery(Categoría.class, "id == 1");  
    Collection<Categoría> cat = (Collection<Categoría>) query.execute();  
    Categoría c = cat.stream().findFirst().get();  
    c.nombre("otro nombre...");  
}  
...
```

Y el *persistent context* se encarga de persistirlo en forma transparente 😊

¿CÓMO USO ZOODB?

QUERIES

```
...
try {
    Query query = pm.newQuery(Producto.class, "precio <= 3000");
    Collection productos = (Collection) query.execute();
    for (Producto producto : productos) {
        System.out.println(producto);
    }
}
...
...
```

Comparadores: !=, ==, <, >, <=, >=, &&, ||. Solo Strings y Números (ya que no todo JDO esta implementado en ZooDB).

¿CÓMO USO ZOODB?

zooActivateRead() y zooActivateWrite()

```
public class Producto extends ZooPC {  
    private Long id;  
    private String nombre;  
    private Categoria categoria;  
    public void categoria(Categoria categoria) {  
        zooActivateWrite();  
        this.categoria = categoria;  
    }  
    public Categoria categoria() {  
        zooActivateRead();  
        return categoria;  
    }  
}
```

Speaker notes

De modo de poder obtener los beneficios del Cascade y los Proxies, los objetos persistentes en sus métodos de lectura y escritura deben utilizar `zooActivateRead()` y `zooActivateWrite()`

