



Linkia FP

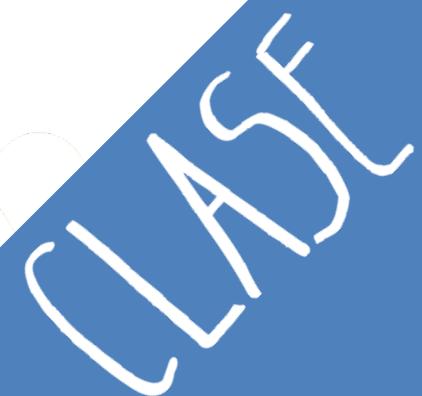
Formación Profesional Oficial a Distancia



DAM – M06 – Clase 08

Acceso a datos

BDOO y BDOR - db4o

CUASF

Contenido del módulo

- UF2: Manejo de conectores.
 - JDBC (mySQL).
 - ORM (Hibernate).
 - BDOO.
 - BDOR.

Temas 2, 3 y 4

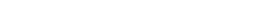
6 Clases

2 Actividades

Plan de trabajo

Plan de trabajo - Curso 2021/2022

DAM - M06: Acceso a datos



UF	FECHAS	TEMAS	Recursos complementarios: Video clases	EVALUACIONES				EXAMEN	
				ACTIVIDADES DE EVALUACIÓN CONTÍNUA					
				Entregas o participaciones 70%	Clases 10%*	Test 20%			
UF1: Persistencia en ficheros	02/02/2022 - 14/02/2022	Tema 1. Persistencia en ficheros	VC01	Actividad 1	C01 C02	Test UF1	Examen UF1		
	14/02/2022 - 28/02/2022	Tema 2. Persistencia en BDR con JDBC	VC02		C03 C04	Test UF2	Examen UF2		
	28/02/2022 - 14/03/2022	Tema 3. Persistencia BDR con ORM	VC03	Actividad 2	C05 C06				
	14/03/2022 - 28/03/2022	Tema 4. Persistencia en BDOO – BDOR	VC04	Actividad 3	C07 C08				
UF3: Persistencia en BD Nativas XML	28/03/2022 - 19/04/2022	Tema 5. Persistencia en BBDD nativas XML	VC05	Actividad 4	C09 C10 C11	Test UF3	Examen UF3		
UF4: Componentes de acceso a datos	19/04/2022 - 02/05/2022	Tema 6. Componentes da acceso a datos	VC06	Actividad 5	C12 C13	Test UF4	Examen UF4		
	02/05/2022 - 16/05/2022				C14				

Horario de las clases

NÚMERO CLASE	DÍA DE LA SEMANA	FECHA	HORA INICIO
C01	MARTES	08/02/2022	15:15
C02	VIERNES	11/02/2022	14:30
C03	VIERNES	18/02/2022	14:30
C04	VIERNES	25/02/2022	14:30
C05	VIERNES	04/03/2022	14:30
C06	VIERNES	11/03/2022	14:30
C07	VIERNES	18/03/2022	14:30
C08	VIERNES	25/03/2022	14:30
C09	VIERNES	01/04/2022	14:30
C10	MARTES	05/04/2022	15:15
C11	VIERNES	08/04/2022	14:30
C12	VIERNES	22/04/2022	14:30
C13	VIERNES	29/04/2022	14:30
C14	VIERNES	06/05/2022	14:30

Contenido clase

UF2: Manejo de conectores.

- Características de las BBDD objeto-relacionales.
- Db4o una alternativa *open source*.

CLASE

Características BBDD-OR

- El término BBDD Objeto-Relacional, se usa para describir una BBDD que ha evolucionado desde el modelo relacional hasta una BBDD híbrida, que contiene la tecnología relacional y la OO.
- Un tipo de objeto representa una entidad del mundo real y se compone de:
 - Nombre.
 - Atributos.
 - Métodos.

CLASE

Creación de objetos

- Para crear un objeto con Oracle utilizamos CREATE TYPE.
- Ejemplo:

```
CREATE TYPE persona AS OBJECT (
    nombre VARCHAR2(30),
    telefono VARCHAR2(20),
);
```

- En ANSI SQL99 sería:

```
define type persona:
  Tuple [nombre:string, telefono:string]
```

Gestión de objetos

- Referencias:
 - Las relaciones entre objetos se establecen mediante columnas o atributos de tipo REF.
- Ejemplo:

```
CREATE TABLE Departamento  
  (NomDept VARCHAR(30), jefe REF persona);
```

- Métodos:
 - Los métodos son procedimientos que se declaran en la definición de un tipo de objeto para dotarlo de comportamiento.

Db4o

- Db4o (Data Base For Objects) es una implementación OODB (Bajo GPL, código abierto) que permite de manera sencilla hacer persistentes objetos creados con Java y recuperarlos posteriormente en otra ejecución.
- Db4o NO es exactamente una BBDD OO que sigue ODMG ya que, entre otras cosas, no implementa un lenguaje ODL.

Ventajas Db4o

- Mayor velocidad de desarrollo (transparencia).
- Mejor rendimiento con objetos de negocio complejos:
 - Árboles, estructuras anidadas, relaciones N a N, relaciones recursivas.
- Fácil backup (la BBDD completa está en un único archivo).

Ventajas Db4o

- No necesita administración.
- Las búsquedas se hacen directamente usando objetos.
- Los cambios en los objetos (agregar o quitar atributos a una clase) se aplican directamente en la BBDD, sin tener que migrar datos ni reconfigurar nada.
- Multiplataforma.

Inconvenientes Db4o

- No existe un lenguaje de consultas como sql (se deben realizar de forma programada).
- No existen restricciones, deben programarse (No implementa integridad referencial).
- Tamaño limitado de los ficheros de BBDD (2GB – 264GB).

Crear POJO

- Primero crearemos la clase POJO cuyos objetos haremos persistentes con db4o.

```
public class Pilot {  
    private String name;  
    private int points;  
  
    public Pilot(String name,int points) {  
        this.name=name;  
        this.points=points;  
    }  
  
    public int getPoints() {  
        return points;  
    }  
}
```

```
public void addPoints(int points) {  
    this.points+=points;  
}  
  
public String getName() {  
    return name;  
}  
  
public String toString() {  
    return name+"/"+points;  
}  
}
```

Abrir la BBDD

- Para acceder al fichero db4o o crear uno nuevo, utilizamos el método *DB4oEmbedded.openFile()*.
 - El primer parámetro es una plantilla de configuración.
Para ello utilizamos :
DB4oEmbedded.newConfiguration()
 - El segundo parámetro es el path al archivo.
- Con esto obtenemos una instancia de ObjectContainer, que representa a la interfaz db4o.

Abrir la BBDD

```
ObjectContainer db = Db4oEmbedded.openFile(Db4oEmbedded  
    .newConfiguration() , DB4OFILENAME) ;  
  
try {  
    // do something with db4o  
} finally {  
    db.close();  
}
```

- Para cerrar el objeto *ObjectContainer* utilizamos el método *close()*.

Guardar la BBDD

- Para guardar un objeto, utilizamos el método *store()* pasándole el objeto a guardar.

```
// storeFirstPilot  
  
Pilot pilot1 = new Pilot("Michael Schumacher", 100);  
db.store(pilot1);  
System.out.println("Stored " + pilot1);
```

Obtener objetos

- Db4o proporciona mecanismos para hacer una query a la BBDD. Uno de ellos es *Query-By-Example*.
- Con este sistema creamos un objeto prototipo para db4o que actúa como plantilla de los objetos que queremos obtener de la BBDD.
- Db4o devolverá todos los objetos que concuerden con el prototipo. Devuelve una instancia *ObjectSet*.

Obtener objetos

```
Pilot proto = new Pilot(null, 0);  
ObjectSet result = db.queryByExample(proto);  
listResult(result);
```

```
public static void listResult(ObjectSet result) {  
    System.out.println(result.size());  
    while(result.hasNext()) {  
        System.out.println(result.next());  
    }  
}
```

- Los valores null i 0 que pasamos al objeto prototipo, no son los valores a buscar, son los valores por defecto.

Obtener objetos

- El método *queryByExample()* también permite obtener todos los objetos de una determinada clase.

```
ObjectSet result = db.queryByExample(Pilot.class);  
listResult(result);
```

- También podemos utilizar el método *query()*.

```
List <Pilot> pilots = db.query(Pilot.class);
```

Obtener objetos

- Para realizar la query en función del valor de uno de los atributos del objeto POJO, tenemos que crear el prototipo con el valor a buscar.

```
Pilot proto = new Pilot("Michael Schumacher", 0);  
ObjectSet result = db.queryByExample(proto);  
listResult(result);
```

Actualizar objetos

- Actualizar objetos es tan sencillo como guardarlos.
- Utilizamos el método *store()* después de haber modificado el objeto POJO.

```
ObjectSet result = db
    .queryByExample(new Pilot("Michael Schumacher", 0));
Pilot found = (Pilot) result.next();
found.addPoints(11);
db.store(found);
System.out.println("Added 11 points for " + found);
retrieveAllPilots(db);
```

Eliminar objetos

- Para eliminar objetos utilizamos el método *delete()*.

```
ObjectSet result = db
    .queryByExample(new Pilot("Michael Schumacher", 0));
Pilot found = (Pilot) result.next();
db.delete(found);
System.out.println("Deleted " + found);
retrieveAllPilots(db);
```

Actividad 03

Base de datos con BDOR-BDOO.

- Fecha de entrega: 06/04/2021
- Proyecto creado en NetBeans
comprimido en .zip

CLASE

Final UF2

- Actividad
 - Fecha de entrega: 28/03/2022
- Preguntas evaluación de clase
- Test



Linkia FP

Formación Profesional Oficial a Distancia

