EJERCICIO RESUELTO
Módulo: Programación

Almacenamiento de objetos con JDO

Descripción:

Existen otros métodos de almacenamiento de Objetos que podemos usar además de Db4o.

En este ejercicio vamos a trabajar con unas librerías JDO que permiten gestionar y manipular objetos que se guardarán en un fichero.

Esta nueva forma de almacenar datos nos permitirá, además, realizar consultas sobre los datos almacenados para poder obtenerlos de forma sencilla y rápida.

Objetivos:

- Entender el uso de JDO
- Aplicar los conocimientos de JDO para crear un espacio de almacenamiento de datos

Recursos:

- Acceso a Internet.
- Netbeans
- Librería ObjectDB

Resolución:

Tarea 1: Crear una aplicación que permita almacenar en disco, usando ObjectDB, un conjunto de datos.

En este ejercicio, vamos a usar ObjectDB como sistema de almacenamiento de datos de forma persistente. Por este motivo, debemos empezar descargando la librería: ObjectDB (https://www.objectdb.com/download)

Una vez descargado y descomprimido el fichero, añadimos la librería como dependencia de nuestra aplicación.

Ahora generamos una nueva clase en la que almacenaremos información.

Serie.java:

```
/*
 * To change this license header, choose License Headers in
 * Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Modelos;

import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

/**
 *
 * @author Juan Iruela
 */
@Entity
public class Serie implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id @GeneratedValue
    private long id;

    private String nombre;
    private double valoracion;

    public Serie() {
    }

    public Serie(String nombre, double valoracion) {
        this.nombre = nombre;
        this.valoracion = valoracion;
    }

    public String getNombre() {
        return nombre;
    }
```

```
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public double getValoracion() {
        return valoracion;
    }

    public void setValoracion(double valoracion) {
        this.valoracion = valoracion;
    }

    public long getId() {
        return id;
    }

    @Override
    public String toString() {
        return String.format(
            "%d: Serie: '%s' - Valoración: %.1f",
            this.id, this.nombre, this.valoracion
        );
    }
}
```

En esta clase hay que destacar varios elementos:

- Implements Serializable: Al ser una clase que nos servirá como almacenamiento en disco, debemos implementar la interfaz serializable.
- @Entity: Define que esta clase se utilizará como entidad en nuestra base de datos.
- @Id: Identifica que el campo que estamos definiendo a continuación será el campo identificador dentro de la entidad.
- @GeneratedValue: Especifica que el sistema generará de forma automática el identificador.

Implantamos la clase principal:

```
/*
 * To change this license header, choose License Headers in
 * Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package psp_u4_s3_e1;

import Modelos.Serie;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;
import javax.persistence.TypedQuery;

/**
```

```
 *
 * @author Juan Iruela
 */
public class PSP_U4_S3_E1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        PSP_U4_S3_E1 programa = new PSP_U4_S3_E1();
        programa.iniciar();
    }

    private void iniciar() {
        // Creamos una conexión a "base de datos"
        // En caso de no existir el fichero, se creara
        EntityManagerFactory entityManagerFactory =
            Persistence.createEntityManagerFactory("series.odb");
        EntityManager entityManager =
            entityManagerFactory.createEntityManager();

        // Guardamos unas cuantas series
        this.insertarSeries(entityManager);

        // Total series en la base de datos:
        Query q1 = entityManager.createQuery(
            "SELECT COUNT(serie) FROM Serie serie");
        System.out.println("Total series: " +
            q1.getSingleResult());

        // Encontrar la valoración media de las series:
        Query q2 = entityManager.createQuery(
            "SELECT AVG(serie.valoracion) FROM Serie serie");
        System.out.println("Valoración media: " +
            q2.getSingleResult());

        // Recuperar todas las series de la base de datos:
        TypedQuery<Serie> consulta = entityManager.createQuery(
            "SELECT serie FROM Serie serie", Serie.class);
        List<Serie> resultados = consulta.getResultList();
        for (Serie serie : resultados) {
            System.out.println(serie);
        }

        // Close the database connection:
        entityManager.close();
        entityManagerFactory.close();
    }

    private void insertarSeries(EntityManager entityManager) {

        entityManager.getTransaction().begin();
        entityManager.persist(new Serie("Los Soprano", 8.80));
        entityManager.persist(new Serie("The wire: bajo escucha",
            9.50));
        entityManager.persist(new Serie("Breaking Bad", 9.80));
        entityManager.persist(new Serie("Mad Men", 7.60));
        entityManager.persist(new Serie("Seinfeld", 9.10));
        ...
        entityManager.getTransaction().commit();
```

```
    }
}
```