Actividad 1.7 Tratamiento de ficheros VI - CRUD + DAO + Properties (MapaMundi)

Marcos Alcalde

A. Investigación sobre el uso de los ficheros “properties” de configuración.

1. Cuatro formas de guardar constantes en un proyecto Java plano (sin frameworks ni Maven)

Constantes en el propio código: Declarar los valores fijos directamente en una clase usando public static final.

Ficheros .properties: Usar un archivo de texto plano externo para guardar las claves y valores (como las rutas o separadores).

Ficheros XML: Emplear archivos XML para organizar configuraciones de forma estructurada.

Variables de entorno o propiedades de la JVM: Pasar los valores desde el propio sistema operativo o al arrancar la máquina virtual de Java.

2. ¿Qué ventajas tiene usar un archivo .properties frente a escribir valores directamente como constantes en el código fuente?

Cambio de entorno: Facilita cambiar de un ordenador de pruebas a uno de producción simplemente cambiando el archivo de configuración.

Código más limpio: Evita tener valores fijos y repartidos por las clases del proyecto.

3. ¿Qué tipo de valores son adecuados para guardarse en un archivo .properties, y cuáles no? ¿Por qué?

Inadecuados: Contraseñas sin seguridad, datos muy grandes o estructuras complejas (como listas u objetos anidados), ya que este archivo solo lee texto plano lineal.

4. ¿Qué limitaciones tiene el uso de archivos .properties y qué otras alternativas modernas existen (por ejemplo, YAML, JSON, bases de datos o Spring @Value)?

Limitaciones: No permite hacer estructuras con jerarquías, todo se lee como texto (tienes que convertir los números o booleanos a mano) y es muy básico.

Alternativas modernas:

JSON o YAML: Mucho más limpios para guardar datos organizados en niveles o estructuras complejas.

Bases de datos: Si la configuración cambia constantemente en tiempo de ejecución.

Spring Boot (@Value): Herramientas de frameworks que cargan y convierten los datos automáticamente.

5. ¿Cómo cargar desde Java un archivo .properties y acceder a sus valores? ¿Qué ocurre si una clave no existe?

Cómo se hace: Se usa la clase Properties, se abre el fichero con un FileInputStream y se carga con el método prop.load(). Para leer un valor se usa prop.getProperty("nombre_clave").

Si la clave no existe: El método devuelve null. Si no se controla en el código, puede provocar un error de tipo NullPointerException. Para evitarlo, se puede poner un valor por defecto usando prop.getProperty("clave", "valor_por_defecto").