

2.3. Ficheros de intercambio

Acceso a Datos

Alejandro Roig Aguilar

alejandro.roig@iesalvarofalomir.org

IES Álvaro Falomir

Curso 2023-2024

Formatos de intercambio

El problema de la comunicación entre sistemas con independencia de su sistema operativo y el lenguaje de programación utilizado para escribir su software ha marcado el mundo de la informática desde sus orígenes.

Los **formatos de intercambio** de datos son un **conjunto de reglas** que definen cómo se **estructuran y organizan los datos** para su intercambio entre diferentes sistemas.

Estos formatos permiten que los datos se transfieran y compartan de manera eficiente y segura, independientemente de la plataforma o lenguaje de programación utilizado. Destacan **CSV**, **XML** y **JSON**.

CSV

El formato **CSV** (**Comma-separated values**) se utiliza para representar **datos tabulares**.

Cada **línea** del archivo representa una **fila de datos**, y cada **campo** de la fila está **separado por un delimitador**, generalmente una coma o un punto y coma.

Son más fáciles de transmitir y ocupan mucho **menos espacio** que los ficheros XML o JSON.

```
id,nombre,ciclo,curso,profesor  
ad,Acceso a Datos,DAM,2,Alejandro  
psp,Programación de servicios y procesos,DAM,2,Juan  
pro,Programación,DAW,1,David
```

XML

El formato **XML** (eXtensible **Markup** Language) **usa etiquetas** para darle significado a los valores que almacenan. Forman una estructura de árbol donde los elementos pueden tener **contenido textual y atributos**.

```
<modulos>
  <modulo id="ad">
    <nombre>Acceso a Datos</nombre>
    <ciclo>DAM</ciclo>
    <curso>2</curso>
    <profesor>Alejandro</profesor>
  </modulo>
  ...
</modulos>
```

JSON

El formato **JSON** (**JavaScript Object Notation**) se utiliza para representar datos estructurados como pares clave-valor de forma jerárquica.

A diferencia de los XML, los JSON tienden a ser **más pequeños**, siendo populares en el **intercambio de datos en línea**.

```
{“modulos”:[{“id”:“ad”,“nombre”:“Acceso a Datos”,  
              “ciclo”:“DAM”, “curso”:2,“profesor”:“Alejandro”}  
             {“id”:“psp”,“nombre”:“Programación de servicios y procesos”,  
              “ciclo”:“DAM”, “curso”:2,“profesor”:“Juan”}  
             {“id”:“pro”,“nombre”:“Programación”,  
              “ciclo”:“DAW”, “curso”:1,“profesor”:“Ana”}]]}
```

CSV

Lectura de ficheros CSV

Files.lines en java.nio

```
List<List<String>> libros = Files.lines(Paths.get("libros.csv"))  
    .map(linea -> Arrays.asList(linea.split(COMMA_DELIMITER)))  
    .toList();
```

Lectura de ficheros CSV

BufferedReader en java.io

```
List<List<String>> libros = new ArrayList<>();  
  
try (FileReader fr = new FileReader("libros.csv");  
    BufferedReader br = new BufferedReader(fr)) {  
    String linea;  
    while ((linea = br.readLine()) != null) {  
        String[] libro = linea.split(COMMA_DELIMITER);  
        libros.add(Arrays.asList(libro));  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}
```


Lectura de ficheros CSV

Librería OpenCSV

```
List<List<String>> libros = new ArrayList<>();  
  
try (FileReader fr = new FileReader("libros.csv");  
    CSVReader csvReader = new CSVReader(fr)) {  
    String[] libro;  
    while ((libro = csvReader.readNext()) != null) {  
        libros.add(Arrays.asList(libro));  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

XML

XML

Los archivos **XML** (eXtensible **Markup** Language) permiten el intercambio de información entre aplicaciones usando un fichero de texto plano al que se le pueden añadir **etiquetas** para darle significado a los valores que almacenan.

Un fichero XML se divide en dos partes:

- **prolog**: metadatos administrativos (declaración XML, procesamiento opcional tipo de documento...)
- **body**: se compone de dos partes: estructural y de contenido

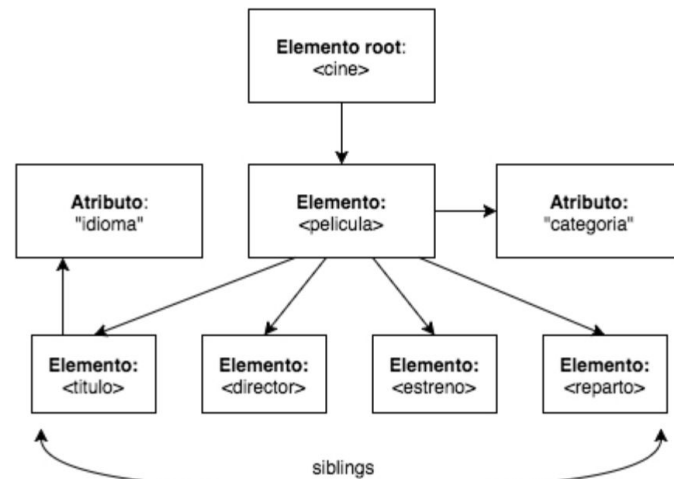
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <cine>
3    <pelicula categoria="accion">
4      <titulo idioma="ingles">Mad Max</titulo>
5      <director>George Miller</director>
6      <estreno>15 mayo 2015</estreno>
7      <reparto>Tom Hardy</reparto>
8      <reparto>Charlize Theron</reparto>
9      <reparto>Nicholas Hoult</reparto>
10   </pelicula>
11   <pelicula categoria="animacion">
12     <titulo idioma="ingles">Inside Out</titulo>
13     <director>Pete Docter</director>
14     <estreno>17 junio 2015</estreno>
15     <reparto>Amy Poehler</reparto>
16     <reparto>Phyllis Smith</reparto>
17     <reparto>Bill Hader</reparto>
18   </pelicula>
19 </cine>
```

XML

Los archivos XML forman una **estructura de tipo árbol**, comenzando desde la raíz (**root**), con ramas (**branches**) hacia las hojas (**leaves**).

Se emplean los términos **parent**, **child** y **sibling** para determinar parentesco: padre, hijo y hermano.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <cine>
3   <pelicula categoria="accion">
4     <titulo idioma="ingles">Mad Max</titulo>
5     <director>George Miller</director>
6     <estreno>15 mayo 2015</estreno>
7     <reparto>Tom Hardy</reparto>
8     <reparto>Charlize Theron</reparto>
9     <reparto>Nicholas Hoult</reparto>
10  </pelicula>
11  <pelicula categoria="animacion">
12    <titulo idioma="ingles">Inside Out</titulo>
13    <director>Pete Docter</director>
14    <estreno>17 junio 2015</estreno>
15    <reparto>Amy Poehler</reparto>
16    <reparto>Phyllis Smith</reparto>
17    <reparto>Bill Hader</reparto>
18  </pelicula>
19 </cine>
```



XML

Existen varias **librerías** para manipular ficheros XML desde Java:

- **DOM** (Document Object Model). Carga el árbol del fichero **en memoria**. Trabajamos nodo a nodo. Recomendable para ficheros pequeños.
- **SAX** (Simple API for XML). No carga el fichero en memoria, sino que **recorre** el fichero de forma **secuencial** informando de la ocurrencia de eventos, como el comienzo de un elemento XML o el final del fichero.
- **JDOM**: Nos permite leer, editar y escribir XML muy fácilmente.
- **JAXB** (Java Architecture for XML Binding). Define como los **objetos** son convertidos en Java desde/a XML.

JAXB

JAXB permite mapear clases Java a representaciones en XML y viceversa.

Proporciona dos **características** principales

- Serialización (**marshalling**) de objetos Java a XML.
- Deserialización (**unmarshalling**) de XML a objetos Java.

Es decir, JAXB permite almacenar y recuperar datos en memoria en cualquier formato XML, sin la necesidad de implementar un conjunto específico de rutinas XML de carga y salvaguarda para la estructura de clases del programa.

JSON

JSON

Los archivos **JSON** (**JavaScript Object Notation**) es un formato ligero para el intercambio de datos de forma estructurada.

Cada vez más popular en lugares donde antes se empleaba XML.

- **Objeto**: Un objeto JSON es una colección de pares nombre : valor
- **Array**: Un array JSON es una colección ordenada de valores separados por “,”

```
[  
  {  
    "nombre": "Pepito Conejo",  
    "edad": 25,  
    "carnet de conducir": true  
  },  
  {  
    "nombre": "Ana Barberá",  
    "edad": 90,  
    "carnet de conducir": false  
  }  
]
```


JSON vs XML

JSON y XML son:

- Autodescriptivos (legibles por un humano)
- Jerárquico (valores dentro de valores)
- Parseables por muchos lenguajes

A diferencia de XML, JSON

- No usa etiquetas
- Es más corto
- Más rápido de leer y escribir
- No usa etiquetas

XML es más difícil de parsear que JSON!

JSON

Existen varias **librerías** para manipular ficheros JSON desde Java. Las principales son:

- **Jackson**
 - <https://github.com/FasterXML/jackson>
 - Es la biblioteca por defecto en **Spring** para JSON.
- **GSON**
 - <https://github.com/google/gson>
 - Biblioteca de JSON de **Google**. Es más ligera que Jackson.