Persistència de dades en XML i JSON

persistència d'objectes estructurada i interoperable

1 Conceptes bàsics

Format	Tipus	Objectiu	Avantatge principal
XML (eXtensible Markup Language)	Text estructurat	Emmagatzemar dades amb jerarquia i metadades	Llegible per humans i màquines; compatible amb molts llenguatges
JSON (JavaScript Object Notation)	Text estructurat	Representar objectes de manera ràpida i lleugera.	Més compacte i senzill; molt usat en APIs i web

2 Representació d'un objecte

Exemple d'objecte `Alumne`:

Java:

```
class Alumne {
   String nom;
   double nota;
}
```

En XML:

```
<alumne>
   <nom>Anna</nom>
   <nota>8.5</nota>
</alumne>
```

EN JSON:

```
{
  "nom": "Anna",
  "nota": 8.5
}
```

*- Observa que **XML** usa etiquetes d'obertura i tancament (`<nom>...</nom>`)*

*- **JSON** utilitzaa claus (`{}`) i parelles clau: valor.*

Persistència en XML en Java

Hi ha diverses formes, però les més clares són:

Hi ha dues maneres principals:

A) Nivell baix: DOM / SAX / StAX

- **DOM (Document Object Model):**
    - Carrega tot el fitxer XML a memòria com un arbre d'elements.
    - Pots afegir, eliminar o modificar nodes.
    - Ideal per fitxers petits o mitjans.
- **SAX / StAX:**
    - Llegeix el fitxer seqüencialment (com un flux).
    - Més ràpid i eficient amb fitxers grans.
    - Només lectura, no permet modificar fàcilment.

B) Nivell alt: JAXB (Java Architecture for XML Binding)

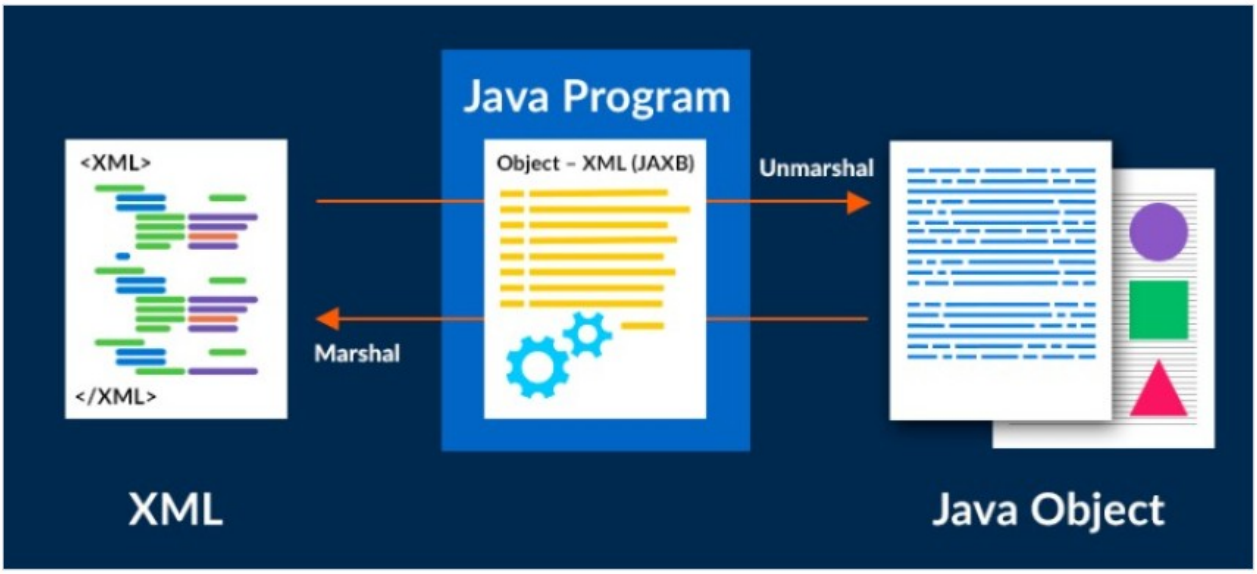
JAXB permet convertir automàticament objectes Java ↔ XML amb molt poc codi.

Exemple:

```
import jakarta.xml.bind.annotation.*; // o javax.xml.bind.annotation.* segons JDK

@XmlRootElement
class Alumne {
   public String nom;
   public double nota;

   public Alumne() {}  // Constructor buit obligatori per JAXB
   public Alumne(String nom, double nota) {
      this.nom = nom;
      this.nota = nota;
   }
}

// Serialitzar:

JAXBContext context = JAXBContext.newInstance(Alumne.class);
Marshaller marshaller = context.createMarshaller();
marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
marshaller.marshal(new Alumne("Anna", 8.5), new File("alumne.xml"));
```

Deserialitzar:

```
Unmarshaller unmarshaller = context.createUnmarshaller();
Alumne a = (Alumne) unmarshaller.unmarshal(new File("alumne.xml"));
System.out.println(a.nom + " té una nota de " + a.nota);
```



4 Persistència en JSON en Java

Exemples amb Gson:

Biblioteques habituals:

- **Gson** (lleugera i senzilla)
- **Jackson** (més completa)
- **org.json** (bàsica)

Gson és la més pràctica.

Exemple amb Gson:

```
import com.google.gson.Gson;
import java.io.*;

class Alumne {
    String nom;
    double nota;

    public Alumne(String nom, double nota) {
        this.nom = nom;
        this.nota = nota;
    }
}

public class ExempleJSON {
    public static void main(String[] args) throws IOException {
        Gson gson = new Gson();

        // Objecte → JSON
```

```
        Alumne a = new Alumne("Anna", 8.5);
        try (FileWriter fw = new FileWriter("alumne.json")) {
            gson.toJson(a, fw);
        }

        // JSON → Objecte
        try (Reader r = new FileReader("alumne.json")) {
            Alumne alumne = gson.fromJson(r, Alumne.class);
            System.out.println(alumne.nom + " nota: " + alumne.nota);
        }
    }
}
```

Important:

- Gson automàticament detecta els camps públics.
- Els fitxers són **text pla**, pots obrir-los i veure-hi el contingut.
- Pots guardar una **llista d'objectes** simplement fent `gson.toJson(llista, writer)`.

5 Diferències pràctiques

Aspecte	XML	JSON
Sintaxi	Etiquetes jeràrquiques	Claus i valors
Llegibilitat	Verbós però clar	Més compacte i modern
Ús habitual	Configuracions, sistemes empresarials	APIs, aplicacions web i mòbils
Llibreria típica	JAXB	Gson / Jackson
Comentaris	Admet `<!-- comentari -->`	No admet comentaris

Bones pràctiques:

- Sempre crear un **constructor buit** per als objectes que es volen serialitzar.
- Controlar possibles **errors d'entrada/sortida (IOException)** amb `try-catch`.
- Format clar i **indentat** (`Marshaller.JAXB_FORMATTED_OUTPUT = true`).
- Tancar correctament els fluxos (`try-with-resources`).
- No sobreescriure fitxers sense confirmar-ho (millor preguntar abans)

- ***XML*** *→ estructurat, ideal per interoperabilitat i dades complexes.*

- **JSON** → lleuger, fàcil de treballar i molt comú a la web.

- **JAXB i Gson** → eines modernes i senzilles per convertir objectes Java en dades persistents.

Serialització i deserialització pas a pas

La **serialització** és el procés de convertir un objecte Java en un format que es pugui **guardar** (fitxer, base de dades, xarxa).
La **deserialització** és el procés invers: **llegir** el fitxer i reconstruir l'objecte Java original.

A la pràctica, això vol dir que podem **guardar objectes (o llistes d'objectes)** en fitxers XML o JSON i recuperar-los fàcilment.

import jakarta.xml.bind.; // o javax.xml.bind segons la versió*

```
import jakarta.xml.bind.*;  // o javax.xml.bind segons la versió
import java.io.*;
import java.util.*;

@XmlRootElement
class Alumne {
    public String nom;
    public double nota;

    // Constructor buit (necessari per JAXB)
    public Alumne() {}

    public Alumne(String nom, double nota) {
        this.nom = nom;
        this.nota = nota;
    }

    @Override
    public String toString() {
        return nom + " (" + nota + ")";
    }
}
```

```
public class ExempleJAXB {
    public static void main(String[] args) throws Exception {
        List<Alumne> alumnes = new ArrayList<>();
        alumnes.add(new Alumne("Anna", 8.5));
        alumnes.add(new Alumne("Marc", 6.2));

        // --- Serialitzar a XML ---
        JAXBContext context = JAXBContext.newInstance(Alumne.class, ArrayList.class);
        Marshaller m = context.createMarshaller();
        m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
        m.marshal(alumnes, new File("alumnes.xml"));

        // --- Deserialitzar des de XML ---
        Unmarshaller um = context.createUnmarshaller();
        List<?> llista = (List<?>) um.unmarshal(new File("alumnes.xml"));
        System.out.println("Llista recuperada: " + llista);
    }
}
```

JAXB és molt útil quan vols tenir dades en format estructurat i llegible per altres aplicacions.

Exemple complet amb JSON (Gson)

```
import com.google.gson.*;
import java.io.*;
import java.util.*;

class Alumne {
    String nom;
    double nota;

    public Alumne(String nom, double nota) {
```

```
        this.nom = nom;
        this.nota = nota;
    }

    @Override
    public String toString() {
        return nom + " (" + nota + ")";
    }
}

public class ExempleGson {
    public static void main(String[] args) throws IOException {
        Gson gson = new GsonBuilder().setPrettyPrinting().create();

        List<Alumne> alumnes = List.of(
            new Alumne("Anna", 8.5),
            new Alumne("Marc", 6.2)
        );

        // --- Serialitzar a JSON ---
        try (FileWriter fw = new FileWriter("alumnes.json")) {
            gson.toJson(alumnes, fw);
        }

        // --- Deserialitzar des de JSON ---
        try (Reader r = new FileReader("alumnes.json")) {
            Alumne[] llista = gson.fromJson(r, Alumne[].class);
            System.out.println("Llista recuperada: " + Arrays.toString(llista));
        }
    }
}
```

*Gson permet treballar fàcilment amb **llistes i col·leccions**.*
*És ideal per guardar fitxers que després es vulguin llegir des de **altres llenguatges** (Python, JavaScript...).*

Gestió d'errors en la persistència

Quan treballem amb fitxers, poden aparèixer errors comuns.
Per evitar que el programa es tanqui sobtadament, utilitzem blocs **`try-catch`**.

```
try (FileWriter fw = new FileWriter("alumnes.json")) {
    gson.toJson(alumnes, fw);
} catch (IOException e) {
    System.err.println("Error escrivint al fitxer: " + e.getMessage());
}
```

Tipus d'error	Exemple	Solució típica
FileNotFoundException	Obrir un fitxer que no existeix	Comprovar amb `new File("nom").exists()`
IOException	Error de lectura o escriptura	Mostrar missatge i aturar de manera controlada
JsonSyntaxException	Fitxer JSON mal format	Comprovar abans de deserialitzar
JAXBException	Error en convertir objectes XML	Verificar etiquetes i constructors

Bones pràctiques per a la persistència

- Fer servir **try-with-resources** perquè Java tanqui automàticament els fitxers.
- Crear un **mètode reutilitzable** per llegir o escriure objectes (no duplicar codi).
- Escriure **fitxers amb format bonic** (`setPrettyPrinting` a Gson o `JAXB_FORMATTED_OUTPUT` a JAXB).
- Guardar els fitxers dins d'una carpeta del projecte (`data/` o `fitxers/`).
- Comprovar si el fitxer existeix abans de sobreescriure'l.
- No oblidar el **constructor buit** per a classes serialitzables.

Connexió amb el gestor d'alumnes (CSV → XML/JSON)

A partir del **GestorCSVCompleto**, pots adaptar el programa perquè:

- En lloc d'escriure línies CSV, guardi una **llista d'objectes `Alumne`** amb Gson o JAXB.
- Afegeixi noves opcions al menú:

```
6. Exportar a XML
7. Exportar a JSON
8. Importar des de XML/JSON
```

Permeti intercanviar fitxers entre formats.
*Això demostra que XML i JSON són **interoperables** i que la persistència és independent del format.*

Objectiu	Format recomanat	Llibreria Java	Classe clau
Llegibilitat i interoperabilitat	XML	JAXB	`Marshaller` / `Unmarshaller`
Lleugeresa i simplicitat	JSON	Gson	`Gson` / `GsonBuilder`
Manipular fitxers grans	XML	StAX / SAX	`XMLStreamReader`
Integració amb web / APIs	JSON	Gson o Jackson	`Gson` / `ObjectMapper`

Codi Gestor Complet CSV (quan fagis els exercicis proposats pots utilitzar el teu Gestor i realitzar les modificacions sobre ell o aquí tens una mostra del codi que vam utilitzar)

```
package Paquet1;

import java.io.*;
import java.util.*;

public class GestorCSVCompleto {
private static final File FITXER = new File("alumnes.csv");

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
```

```
    int opcio;

do {
System.out.println("\n--- GESTOR D'ALUMNES (CSV) ---");
System.out.println("1. Afegir alumne");
System.out.println("2. Llistar alumnes");
System.out.println("3. Calcular nota mitjana");
System.out.println("4. Cercar alumne pel nom");
System.out.println("5. Eliminar alumne");
System.out.println("0. Sortir");
System.out.print("Opció: ");
opcio = llegirEnter(sc);

switch (opcio) {
case 1 -> afegirAlumne(sc);
case 2 -> llistarAlumnes();
case 3 -> calcularMitjana();
case 4 -> cercarAlumne(sc);
case 5 -> eliminarAlumne(sc);
case 0 -> System.out.println("Sortint...");
default -> System.out.println("Opció no vàlida");
}
} while (opcio != 0);

sc.close();
}

private static void afegirAlumne(Scanner sc) {
System.out.print("Nom: ");
String nom = sc.nextLine();
System.out.print("Cognom: ");
String cognom = sc.nextLine();
```

```
System.out.print("Edat: ");
int edat = llegirEnter(sc);
System.out.print("Nota: ");
double nota = llegirDouble(sc);

try (PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter(FITXER, true)))) {
pw.println(nom + "," + cognom + "," + edat + "," + nota);
System.out.println("Alumne afegit correctament!");
} catch (IOException e) {
System.err.println("Error escrivint: " + e.getMessage());
}
}

private static void llistarAlumnes() {
List<String[]> alumnes = llegirFitxer();
if (alumnes.isEmpty()) {
System.out.println("Encara no hi ha alumnes.");
return;
}

// Ordenem per nota descendent
alumnes.sort((a, b) -> Double.compare(Double.parseDouble(b[3]),
Double.parseDouble(a[3])));

System.out.println("\n--- LLISTA D'ALUMNES ---");
for (String[] alumne : alumnes) {
String nomComplet = alumne[0] + " " + alumne[1];
int edat = Integer.parseInt(alumne[2]);
double nota = Double.parseDouble(alumne[3]);
String estat = (nota >= 5) ? "✅Aprovat" : "❌Suspès";
System.out.println("Nom complet: " + nomComplet +
```

```
" | Edat: " + edat +
" | Nota: " + nota +
" | " + estat);

}
}

private static void calcularMitjana() {
List<String[]> alumnes = llegirFitxer();
if (alumnes.isEmpty()) {
System.out.println("Encara no hi ha alumnes.");
return;
}
double suma = 0;
for (String[] alumne : alumnes) {
suma += Double.parseDouble(alumne[3]);
}
double mitjana = suma / alumnes.size();
System.out.println("La mitjana de notes és: " + mitjana);
}

private static void cercarAlumne(Scanner sc) {
System.out.print("Nom a cercar: ");
String nomBuscat = sc.nextLine().trim();

List<String[]> alumnes = llegirFitxer();
boolean trobat = false;

for (String[] alumne : alumnes) {
if (alumne[0].equalsIgnoreCase(nomBuscat)) {
System.out.println("Trobat -> " + alumne[0] + " " + alumne[1]
+
```

```
" | Edat: " + alumne[2] +
" | Nota: " + alumne[3]);

trobat = true;
}
}

if (!trobat) {
System.out.println("No s'ha trobat cap alumne amb aquest nom.");
}
}

private static void eliminarAlumne(Scanner sc) {

System.out.print("Nom de l'alumne a eliminar: ");
String nomEliminar = sc.nextLine().trim();

List<String[]> alumnes = llegirFitxer();
boolean eliminat = alumnes.removeIf(a ->
a[0].equalsIgnoreCase(nomEliminar));

if (eliminat) {
// Tornem a escriure el fitxer des de zero
try (PrintWriter pw = new PrintWriter(new BufferedWriter(new
FileWriter(FITXER, false)))) {
for (String[] alumne : alumnes) {
pw.println(String.join(",", alumne));
}
} catch (IOException e) {
System.err.println("Error reescrivint: " + e.getMessage());
}
System.out.println("Alumne eliminat correctament!");
```

```
} else {
System.out.println("No s'ha trobat cap alumne amb aquest nom.");
}
}

// --- Utilitats ---

private static List<String[]> llegirFitxer() {
List<String[]> alumnes = new ArrayList<>();
if (!FITXER.exists()) return alumnes;

try (BufferedReader br = new BufferedReader(new FileReader(FITXER))) {
String linia;
while ((linia = br.readLine()) != null) {
String[] parts = linia.split(",");
if (parts.length == 4) alumnes.add(parts);
}
} catch (IOException e) {
System.err.println("Error llegint: " + e.getMessage());
}
return alumnes;
}

private static int llegirEnter(Scanner sc) {
while (true) {
try {
return Integer.parseInt(sc.nextLine().trim());
} catch (NumberFormatException e) {
System.out.print("Introdueix un nombre enter vàlid: ");
}
}
}
```

```
private static double llegirDouble(Scanner sc) {
while (true) {
try {
return Double.parseDouble(sc.nextLine().trim());
} catch (NumberFormatException e) {
System.out.print("Introdueix un nombre decimal vàlid: ");
}
}
}
}
```