

UD08_T02: JSON y YAML



... es el **poder** de la POO ...

1. **Introducción a JSON y YAML**

- 1. 1. ¿Qué es JSON?
- 1. 2. ¿Qué es YAML?
- 1. 3. Comparación entre JSON y YAML:

2. **Generación y Carga de JSON y YAML en Java**

3. **Actividades**

4. **Fuentes de información**

1. Introducción a JSON y YAML

1.1. ¿Qué es JSON?

[JSON](#) (JavaScript Object Notation) es un formato ligero de intercambio de datos que es fácil de leer y escribir para los humanos, y fácil de parsear y generar para las máquinas. Es un formato basado en texto que utiliza una estructura de pares clave-valor, similar a los objetos en JavaScript.

1.2. ¿Qué es YAML?

[YAML](#) (YAML Ain't Markup Language) es un formato de serialización de datos legible por humanos que se utiliza comúnmente para archivos de configuración y en aplicaciones donde los datos deben ser almacenados o transmitidos. YAML es más expresivo que JSON y permite comentarios, lo que lo hace más adecuado para configuraciones complejas.

1.3. Comparación entre JSON y YAML:

Característica	JSON	YAML
Legibilidad	Buena, pero menos expresivo	Muy buena, más expresivo
Comentarios	No soporta comentarios	Soporta comentarios
Estructura	Basado en pares clave-valor	Basado en indentación
Uso común	APIs, intercambio de datos	Configuraciones, DevOps
Complejidad	Más simple	Más flexible y complejo

2. Generación y Carga de JSON y YAML en Java

Para trabajar con JSON y YAML en Java, utilizaremos las bibliotecas **Jackson** (para JSON) y **SnakeYAML** (para YAML).

Dependencias Maven:

```

1  <dependencies>
2    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
databind -->
3    <!-- Dependencia para trabajar con JSON -->
4    <dependency>
5      <groupId>com.fasterxml.jackson.core</groupId>
6      <artifactId>jackson-databind</artifactId>
7      <version>2.15.2</version> <!-- Usa la versión más reciente -->
8    </dependency>
9    <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-
dataformat-yaml -->
10   <!-- Dependencia para trabajar con YAML -->
11   <dependency>
12     <groupId>com.fasterxml.jackson.dataformat</groupId>
13     <artifactId>jackson-dataformat-yaml</artifactId>
14     <version>2.15.2</version> <!-- Usa la versión más reciente -->
15   </dependency>
16 </dependencies>

```

Ejemplo Simple: Generación y Carga de JSON y YAML

Clase `Libro`:

```

1  public class Libro {
2    private String titulo;
3    private String autor;
4    private int anyoPublicacion;
5
6    //Constructor por defecto, necesario para Jackson
7    public Libro() {
8    }
9
10   // Constructor, getters y setters
11   public Libro(String titulo, String autor, int anyoPublicacion) {
12     this.titulo = titulo;
13     this.autor = autor;
14     this.anyoPublicacion = anyoPublicacion;
15   }
16
17   public String getTitulo() {
18     return titulo;
19   }
20
21   public void setTitulo(String titulo) {
22     this.titulo = titulo;
23   }
24
25   public String getAutor() {
26     return autor;
27   }
28
29   public void setAutor(String autor) {
30     this.autor = autor;
31   }
32
33   public int getAnyoPublicacion() {
34     return anyoPublicacion;
35   }
36
37   public void setAnyoPublicacion(int anyoPublicacion) {
38     this.anyoPublicacion = anyoPublicacion;
39   }
40
41   @Override

```

```

42     public String toString() {
43         return "Libro{" +
44             "titulo='" + titulo + '\'' +
45             ", autor='" + autor + '\'' +
46             ", año Publicacion=" + anyoPublicacion +
47             '}' ;
48     }
49 }

```

```

1  import com.fasterxml.jackson.databind.ObjectMapper;
2  import com.fasterxml.jackson.dataformat.yaml.YAMLFactory;
3
4  import java.io.File;
5  import java.io.IOException;
6  import java.util.ArrayList;
7
8  public class TestLibro {
9      public static void main(String[] args) {
10         // Crear una lista de libros
11         ArrayList<Libro> libros = new ArrayList<>();
12         libros.add(new Libro("El Principito", "Antoine de Saint-Exupéry", 1943));
13         libros.add(new Libro("Cien Años de Soledad", "Gabriel García Márquez", 1967));
14         libros.add(new Libro("1984", "George Orwell", 1949));
15
16         // Generar JSON con Jackson
17         ObjectMapper jsonMapper = new ObjectMapper(); // ObjectMapper de Jackson: permite
leer y escribir JSON
18         try {
19             jsonMapper.writeValue(new File("libros.json"), libros);
20         } catch (IOException ex) {
21             System.out.println("Error al generar el archivo JSON.");
22         }
23         System.out.println("Archivo JSON generado con éxito.");
24
25         // Cargar JSON con Jackson
26         ArrayList<Libro> librosCargadosJson = null;
27         try {
28             librosCargadosJson = jsonMapper.readValue(new File("libros.json"),
jsonMapper.getTypeFactory().constructCollectionType(ArrayList.class, Libro.class));
29         } catch (IOException ex) {
30             System.out.println("Error al cargar el archivo JSON.");
31         }
32         System.out.println("Libros cargados desde JSON:");
33         if (librosCargadosJson != null) {
34             for (Libro libro : librosCargadosJson) {
35                 System.out.println(libro);
36             }
37         } else {
38             System.out.println("No se han cargado libros del JSON.");
39         }
40
41         // Generar YAML con Jackson
42         ObjectMapper yamlMapper = new ObjectMapper(new YAMLFactory());
43         try {
44             yamlMapper.writeValue(new File("libros.yaml"), libros);
45         } catch (IOException ex) {
46             System.out.println("Error al generar el archivo YAML.");
47         }
48         System.out.println("Archivo YAML generado con éxito.");
49
50         // Cargar YAML con Jackson
51         ArrayList<Libro> librosCargadosYaml = null;
52         try {
53             librosCargadosYaml = yamlMapper.readValue(new File("libros.yaml"),
jsonMapper.getTypeFactory().constructCollectionType(ArrayList.class, Libro.class));
54         } catch (IOException ex) {
55             System.out.println("Error al cargar el archivo YAML.");
56         }
57         System.out.println("Libros cargados desde YAML:");
58         if (librosCargadosYaml != null) {
59             for (Libro libro : librosCargadosYaml) {
60                 System.out.println(libro);

```

```

61         }
62     } else {
63         System.out.println("No se han cargado libros del YAML.");
64     }
65 }
66 }

```

Resultado Esperado:**Archivo `libros.json`:**

```

1  [
2  {
3      "titulo": "El Principito",
4      "autor": "Antoine de Saint-Exupéry",
5      "anyoPublicacion": 1943
6  },
7  {
8      "titulo": "Cien Años de Soledad",
9      "autor": "Gabriel García Márquez",
10     "anyoPublicacion": 1967
11 },
12 {
13     "titulo": "1984",
14     "autor": "George Orwell",
15     "anyoPublicacion": 1949
16 }
17 ]

```

Archivo `libros.yaml`:

```

1  ---
2  - titulo: "El Principito"
3    autor: "Antoine de Saint-Exupéry"
4    anyoPublicacion: 1943
5  - titulo: "Cien Años de Soledad"
6    autor: "Gabriel García Márquez"
7    anyoPublicacion: 1967
8  - titulo: "1984"
9    autor: "George Orwell"
10   anyoPublicacion: 1949

```

Salida en Consola:

```

1  Archivo JSON generado con éxito.
2  Libros cargados desde JSON:
3  Libro{titulo='El Principito', autor='Antoine de Saint-Exupéry', año Publicacion=1943}
4  Libro{titulo='Cien Años de Soledad', autor='Gabriel García Márquez', año Publicacion=1967}
5  Libro{titulo='1984', autor='George Orwell', año Publicacion=1949}
6  Archivo YAML generado con éxito.
7  Libros cargados desde YAML:
8  Libro{titulo='El Principito', autor='Antoine de Saint-Exupéry', año Publicacion=1943}
9  Libro{titulo='Cien Años de Soledad', autor='Gabriel García Márquez', año Publicacion=1967}
10 Libro{titulo='1984', autor='George Orwell', año Publicacion=1949}

```

3. Actividades

Genera un nuevo proyecto, **crea una clase** `Producto` con los siguientes atributos:

- `nombre` (String)
- `precio` (double)
- `stock` (int)

Crear otra clase `TestProducto` **con una lista de objetos** `Producto` con al menos 3 productos.

Genera y guarda la lista en un archivo JSON (`productos.json`) y en un archivo YAML (`productos.yaml`).

Carga y muestra los datos desde ambos archivos.

Genera un zip con el proyecto de IntelliJ. Envía el archivo zip a la tarea de Aules.

4. Fuentes de información

- <https://awsacademyinstructure.com>
- <https://www.deepseek.com/>