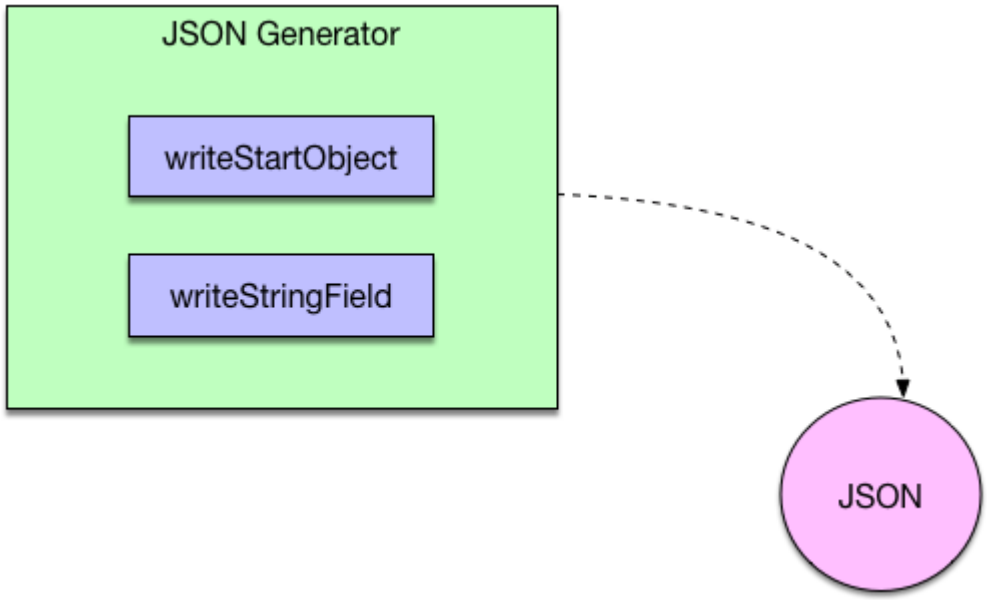
El uso de un JSON Generator es muy común en muchos lenguajes y se basa en una clase o conjunto de clases que sean capaces de generar datos en json de forma sencilla. Vamos a ver como realizar esta tarea en Java . Para ello lo primero va a ser instalar las librerías de Jackson Core en un proyecto Maven.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.arquitecturajava</groupId>
  <artifactId>jsongenerator</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
core -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-core</artifactId>
    <version>2.9.6</version>
</dependency>
  </dependencies>
</project>
```

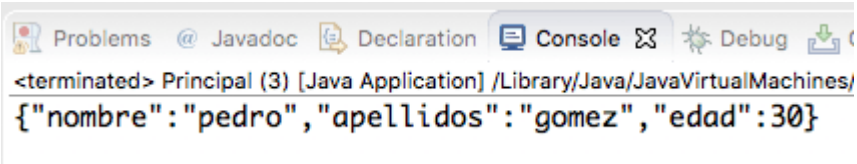
Una vez tenemos las librerías el siguiente paso es construir un JSON Generator que nos permita construir un fichero JSON elemental.

```
package com.arquitecturajava;

import java.io.IOException;

import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonGenerator;

public class Principal {

	public static void main(String[] args) throws IOException {
		JsonFactory factoria= new JsonFactory();
		JsonGenerator generator=
factoria.createGenerator(System.out);
		generator.writeStartObject();
		generator.writeStringField("nombre", "pedro");
		generator.writeStringField("apellidos", "gomez");
		generator.writeNumberField("edad",30);
		generator.writeEndObject();
		generator.flush();

	}

}
```

En este primer ejemplo hemos usado un JsonGenerator partiendo de un JsonFactory y simplemente le estamos solicitando que nos construya un fichero json que contenga el concepto de persona con las propiedades (nombre, apellidos ,edad) .Para ello hemos usado el método writeStartObject , writeStringField



Ademas hemos solicitado que lo imprima en la consola por lo tanto nada mas ejecutar el programa podremos ver como se imprime el objeto .

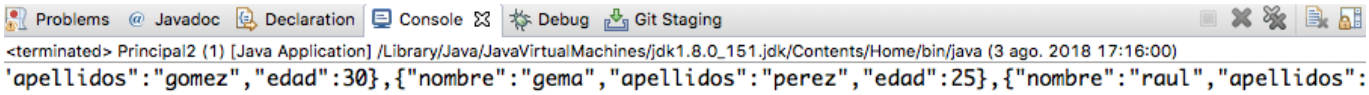


La realidad es que es muy sencillo operar con esta librería. Si necesitamos por ejemplo generar un array que es una de las estructuras más típicas sera suficiente con hacer.

```
package com.arquitecturajava;

import java.io.IOException;

import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonGenerator;
```

```
public class Principal2 {

	public static void main(String[] args) throws IOException {
		JsonFactory factoria= new JsonFactory();
		JsonGenerator generator=
factoria.createGenerator(System.out);
		generator.writeStartArray();
		generator.writeStartObject();
		generator.writeStringField("nombre", "pedro");
		generator.writeStringField("apellidos", "gomez");
		generator.writeNumberField("edad",30);
		generator.writeEndObject();
		generator.writeStartObject();
		generator.writeStringField("nombre", "gema");
		generator.writeStringField("apellidos", "perez");
		generator.writeNumberField("edad",25);
		generator.writeEndObject();
		generator.writeStartObject();
		generator.writeStringField("nombre", "raul");
		generator.writeStringField("apellidos", "gonzalez");
		generator.writeNumberField("edad",40);
		generator.writeEndObject();
		generator.writeEndArray();
		generator.flush();

	}

}
```

Demonos cuenta como hemos usado writeStartArray ademas de writeStartObject. El resultado de igual forma se imprimirá en la consola:



El manejo de JSON es una de las necesidades más habituales en la programación Java hoy en día . Este tipo de librerías son las más sencillas para abordarlo.

Otros artículos relacionados

1. Java JSON utilizan la librería mJson
2. JSON Viewer Awesome y chrome
3. Introducción a JSON Web Token y la seguridad
4. JAX-RS Client y JSON
5. https://www.json.org/