Tabla de Contenidos

- Java JSON y Jackson
- Java JSON y escrituras
- Otros artículos relacionados

Java JSON , las operaciones de lectura y escritura de JSON con Java son unas de las operaciones mas habituales. Vamos a abordar como leer un fichero JSON y como volverle a escribir usando Jackon. Lo primero que tendremos que hacer es crear un proyecto Maven con Eclipse que nos permita añadir la dependencia de JackSon.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>org.example</groupId>
    <artifactId>JavaJSON</artifactId>
    <version>1.0-SNAPSHOT</version>

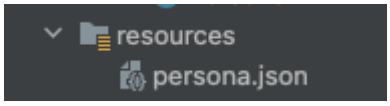
    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>
    <dependencies>

        <!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-
databind -->
```

```
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.13.1</version>
        </dependency>
    </dependencies>

</project>
```

Una vez tenemos este paso realizado el siguiente paso es definir un bloque de código que con Maven nos permita leer un fichero que este en la carpeta de resources.



Este bloque de código nos permite leer el fichero:

```
private File getFileFromResource(String fichero) throws
URISyntaxException {

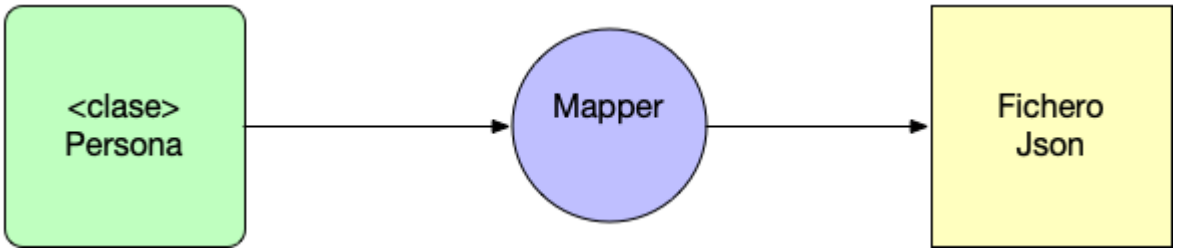
    ClassLoader cargador = getClass().getClassLoader();
    URL resource = cargador.getResource(fichero);
    if (resource == null) {
        throw new IllegalArgumentException("fichero no encontrado" +
fichero);
    } else {

        return new File(resource.toURI());
    }

}
```

Java JSON y Jackson

Es momento de usar Jackson para leer el fichero. Jackson usa lo que se denomina un Object Mapper para mapear el fichero JSON a un Objeto.



Vamos a ver la operación en código:

```
package com.arquitecturajava;

public class Persona {
    private String nombre;
    private String apellidos;
    private int edad;

    public Persona(String nombre, String apellidos, int edad) {
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.edad = edad;
    }

    public Persona() {
    }

    public String getNombre() {
        return nombre;
    }
```

```
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellidos() {
        return apellidos;
    }

    public void setApellidos(String apellidos) {
        this.apellidos = apellidos;
    }

    public int getEdad() {
        return edad;
    }

    public void setEdad(int edad) {
        this.edad = edad;
    }
}
```

En primer lugar tenemos la clase Persona, es momento de construir el programa main.

```
package com.arquitecturajava;

import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.*;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.nio.file.Paths;
```

```
import java.util.List;

public class LectorJSON {
    public static void main(String[] args) throws URISyntaxException {

        LectorJSON lector = new LectorJSON();
        File fichero = lector.getFileFromResource("persona.json");
        System.out.println(fichero);



        ObjectMapper mapper = new ObjectMapper();



        try {

            Persona persona = mapper.readValue(fichero,
Persona.class);
            System.out.println(persona.getNombre());
            System.out.println(persona.getApellidos());
            System.out.println(persona.getEdad());

        } catch (IOException e) {
            e.printStackTrace();
        }



    }

    private File getFileFromResource(String fichero) throws
URISyntaxException {
```

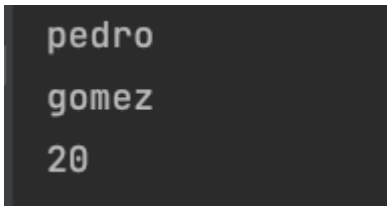
```
        ClassLoader cargador = getClass().getClassLoader();
        URL resource = cargador.getResource(fichero);
        if (resource == null) {
            throw new IllegalArgumentException("fichero no encontrado"
+ fichero);
        } else {

            return new File(resource.toURI());
        }

    }

}
```

Como se puede observar se genera un mapper y se usa el método readValue para convertir el fichero en un objeto en formato JSON una vez lo tenemos cargado lo mostramos por la consola:



Java JSON y escrituras

Es momento de realizar la operación inversa y construir un objeto Persona para que luego nos genere un fichero de JSON.

```
public static void main(String[] args) throws URISyntaxException {

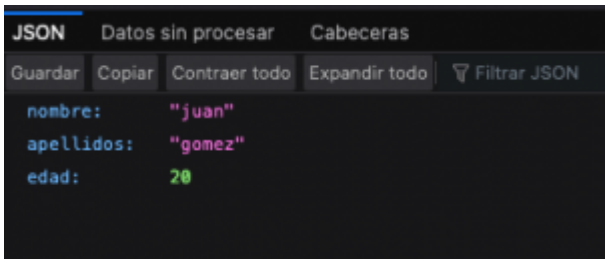
       EscritorJSON lector = new EscritorJSON();
       ObjectMapper mapper = new ObjectMapper();
       Persona p = new Persona("juan", "gomez", 20);
```

```
        try {
            File fichero = new File("persona2.json");
            fichero.createNewFile();
            mapper.writeValue(fichero, p);


        } catch (IOException e) {
            e.printStackTrace();
        }


    }
```

El resultado lo podemos ver en Firefox acabamos de generar información en JSON.



Jackson es una de las librerías de referencia a la hora de manejar JSON ya que es la librería en la que Spring Framework se apoya

Otros artículos relacionados

- Spring Boot REST JPA y JSON
- Servlet JSON y el manejo de Ajax
- JSON API y Arquitecturas REST