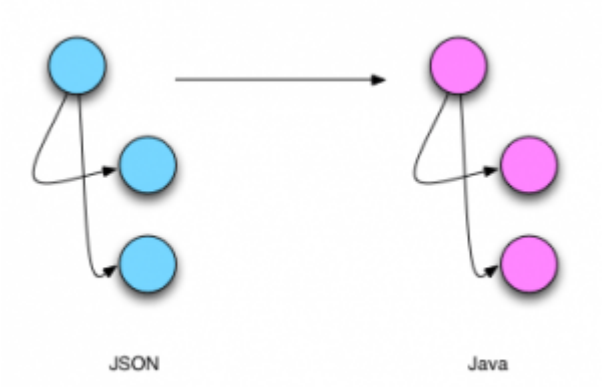
Todos usamos Java JSON en el trabajo diario . JSON ha convertido en el formato de intercambio de información más natural entre distintos tipos de aplicaciones. En Java la mayor parte de las librería realizan un enfoque basado en mapear estructuras JSON a clases Java .



Java JSON

Sin embargo puede que no siempre queramos este enfoque ya que nos obligará a construir un conjunto elevado de nuevas clases Java para nuestra aplicación. mJSON es una librería que nos permite un enfoque más orientado al manejo de la estructura en formato mapa. Eliminando la necesidad de trabajar con un mapeo tan directo y proporcionándonos un API elegante . Vamos a ver un par de ejemplos ,para ello nos crearemos un proyecto Maven que incluya la dependencia.

```
<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>UsarMJSON</artifactId>
```

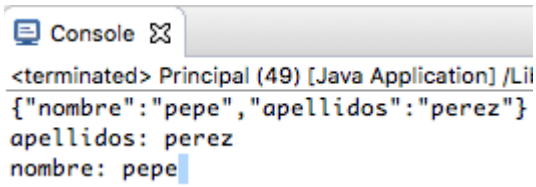
```
        <version>1</version>
        <dependencies>
                <dependency>
                        <groupId>org.sharegov</groupId>
                        <artifactId>mjson</artifactId>
                        <version>1.3</version>
                </dependency>
        </dependencies>
</project>
```

Una vez tenemos la dependencia configurada vamos a ver como podemos recorrer una estructura Json Sencilla usando mJSON.

```
package com.arquitecturajava;
import java.util.Map;

import mjson.Json;

public class Principal {

        public static void main(String[] args) {

                String personaJson=
"{\"nombre\":\"pepe\",\"apellidos\":\"perez\"}";
                System.out.println(personaJson);
                Json miJson= Json.read(personaJson);
                Map<String, Object> propiedades = miJson.asMap();
 for (Map.Entry<String, Object> propiedad: propiedades.entrySet())
 System.out.println(propiedad.getKey() + ": " + propiedad.getValue());
        }
```

```
}
```

Vemos como el resultado se imprime por la consola organizado con las diferentes propiedades.



Acabamos de gestionar una estructura JSON apoyándonos en el concepto de mapa o diccionario. De igual forma podemos modificar la estructura JSON usando este API.

```
package com.arquitecturajava;

import java.util.Map;

import mjson.Json;

public class Principal2 {

	public static void main(String[] args) {

		String personaJson =
"{\"nombre\":\"pepe\",\"apellidos\":\"perez\"}";
		System.out.println(personaJson);

		Json miJson = Json.read(personaJson);
```

```
                miJson.set("nombre", "gema");
                miJson.set("apellidos", "sanchez");

        Map<String, Object> propiedades = miJson.asMap();
 for (Map.Entry<String, Object> propiedad: propiedades.entrySet())
 System.out.println(propiedad.getKey() + ": " + propiedad.getValue());

}
```

De esta forma podemos trabajar de forma sencilla con estructuras JSON sin tener la necesidad de declarar un gran conjunto de clases.

Otros artículos relacionados: JAXRS-Client y JSON , REST JSON y Java ,JavaScript y JSON