https://www.json.org/json-en.html

JSON es un formato que almacena información de forma estructurada. Es un formato muy usado en la actualidad por las aplicaciones informáticas, por ser un formato simple y claro. Es por ello que es interesante poder saber manejar este tipo de información.

Además, en algunas ocasiones, trabajar con objetos puede ser un poco inflexible en el sentido de que un objeto tiene muchos atributos que puede que no necesitemos. La flexibilidad que nos aporta JSON solventa este problema.

Un objeto JSON está normalmente formado por pareja de valores **key:value**, separados por coma.

```
{"nombre":"Juan","edad":19}
```

- key: es una cadena
- value: puede ser una cadena, un numero, un booleano, un array, incluso otro objeto json.

```
{
 "nombre": "Juan",
 "edad": 19,
 "direccion": {
  "numero": 25,
  "calle": "Avda. Juan Carlos I",
  "localidad": "Lorca",
  "provincia": "Murcia",
  "cp": 30800
 }
}
```

Podemos agrupar varios objetos JSON en un array, es algo muy normal cuando devolvemos varios objetos. Por ejemplo:

```
[
    {
          "name": 'Zena
          "sex": 'female',
          "age": 12
    },
    {
          "name": 'Maxwell',
          "sex": 'male',
          "age": 15
    }
]
```

Un uso de JSON es ofrecer en Internet una API, que al consultarla, nos ofrece información que una aplicación puede manejar. Algunas son gratuitas, otras de pago:

https://swapi.dev –> https://swapi.dev/api/films/1

https://dummyjson.com –> ejemplo: https://dummyjson.com/product/1

Otro uso podría ser un tipo de dato a devolver por una consulta a una base de datos. Imagina que en la tabla agenda, formada por los campos nombre, telefono, edad, dirección, en lugar de devolver personas completas, me interesara solo el nombre y la edad. Podríamos crear objetos JSON con solo esos dos campos.

JSON en JAVA: paquete org.json

En Java, existen varias librerías para manejar objetos de tipo json. Una de ellas es la librería o paquete **org.json**. Hay varias versiones disponibles, la última la versión 20230227.

Este paquete lo podemos incluir en nuestro proyecto si usamos Maven cargando las dependencias en el pom.xml o podemos descargarlo en formato .jar e incluirlo en mis proyectos.

- Acceso a la ultima versión –> https://mvnrepository.com/artifact/org.json/json

Si queremos la librería .jar (json-20230227.jar), hay que ir al HomePage en github y descargarlo desde allí. En el archivo README.md hay un enlace.

- Documentación JAVA –> http://stleary.github.io/JSON-java/index.html
- Web con ejemplos de uso –> https://www.baeldung.com/java-org-json

Ejemplos de uso

Crearemos un nuevo proyecto con el gestor de proyectos ANT y le añadiremos a mi proyecto la librería *json-20230227.jar* descargada de Internet.

Los dos tipos de datos que vamos a usar son JSONObject y JSONArray.

Ejemplo1. Creamos un objeto JSON simple y lo imprimimos

```
JSONObject jsonObject;

jsonObject = new JSONObject();
jsonObject.put("nombre", "Juan");
jsonObject.put("edad", 19);
jsonObject.put("email", "juan@kk.com");

System.out.println(jsonObject.toString());
System.out.println(jsonObject.toString(1));

String jsonObjectString = jsonObject.toString(1);
System.out.println(jsonObjectString);
```

```
{"nombre":"Juan","edad":19,"email":"juan@kk.com"}
{
 "nombre": "Juan",
 "edad": 19,
 "email": "juan@kk.com"
}
{
 "nombre": "Juan",
 "edad": 19,
 "email": "juan@kk.com"
}
```

Ejemplo2. Creamos un array con dos objetos JSON

```
JSONObject jo;
JSONArray ja = new JSONArray();

jo = new JSONObject();
jo.put("nombre", "Juan");
jo.put("edad", 19);
jo.put("email", "juan@kk.com");
ja.put(jo);

jo = new JSONObject();
jo.put("nombre", "Pepe");
jo.put("edad", 66);
jo.put("email", "pepe@kk.com");
ja.put(jo);

System.out.println(ja.toString(1));
```

```
[
 {
  "nombre": "Juan",
  "edad": 19,
  "email": "juan@kk.com"
 },
 {
  "nombre": "Pepe",
  "edad": 66,
  "email": "pepe@kk.com"
 }
]
```

Ejemplo 3. JSON anidado.

```
JSONArray ja_listacontactos = new JSONArray();
JSONObject jo_contacto;
JSONObject json_direccion;

//Primer contacto
jo_contacto = new JSONObject();
jo_contacto.put("nombre", "Juan");
jo_contacto.put("edad", 19);
jo_contacto.put("email", "juan@kk.com");

json_direccion = new JSONObject();
json_direccion.put("calle", "Avda. Juan Carlos
I");
json_direccion.put("numero", 25);
json_direccion.put("cp", 30800);
json_direccion.put("localidad", "Lorca");
json_direccion.put("provincia", "Murcia");

jo_contacto.put("direccion",json_direccion);

ja_listacontactos.put(jo_contacto);

//Segundo contacto
jo_contacto = new JSONObject();
jo_contacto.put("nombre", "Pepe");
jo_contacto.put("edad", 66);
jo_contacto.put("email", "pepe@kk.com");
ja_listacontactos.put(jo_contacto);

String datos = ja_listacontactos.toString(1);
System.out.println(datos);
```

```
[
 {
  "direccion": {
   "numero": 25,
   "calle": "Avda. Juan Carlos I",
   "localidad": "Lorca",
   "provincia": "Murcia",
   "cp": 30800
  },
  "nombre": "Juan",
  "edad": 19,
  "email": "juan@kk.com"
 },
 {
  "nombre": "Pepe",
  "edad": 66,
  "email": "pepe@kk.com"
 }
]
```

Ejemplo 4. Recorrido de un array

El método `JSONArray.get(indice)` devuelve el objeto JSON en dicho indice. Hay que hacer un casting porque dicho método devuelve un Object genérico. Otra opción es usar directamente el método `JSONArray.getJSONObject(indice)`

Ojo, que si le pasamos un índice que no existe, salta una excepción. Para que no salte y devuelva null, esta el método `JSONArray.opt(indice)` y `JSONArray.optJSONObject(indice)`

```
//Recorremos el array e imprimimos los objetos que tenga
JSONObject json;
for (int i = 0; i < ja_listacontactos.length(); i++) {

    json = ja_listacontactos.getJSONObject(i);
    //json = (JSONObject)ja_listacontactos.get(i);

    System.out.println("Objeto numero "+i);
    System.out.println("------------------");
    System.out.println(json.toString(1));
    System.out.println("");
}
```

```
Objeto numero 0
------------------
{
 "direccion": {
  "numero": 25,
  "calle": "Avda. Juan Carlos I",
  "localidad": "Lorca",
  "provincia": "Murcia",
  "cp": 30800
 },
 "nombre": "Juan",
 "edad": 19,
 "email": "juan@kk.com"
}

Objeto numero 1
------------------
{
 "nombre": "Pepe",
 "edad": 66,
 "email": "pepe@kk.com"
}
```

Ejemplo 5. Lectura de campos de un JSON

Como hemos explicado antes, hay dos métodos para leer valores, tanto si es un objeto json como si es un array json:

- get(clave/indice): devuelve el valor asociado. Si no existiera, salta una excepción. Este método devuelve elementos de tipo Object, por lo que hay que hacer un casting.

- opt(clave/indice): devuelve el valor asociado o null si no existe. Al igual que antes, devuelve elementos de tipo Object.

- has(clave): indica **true** si existe dicha clave.

Adicionalmente, si no queremos hacer el casting, tenemos los métodos getString(), getJSONObject(), getInt(), etc.

```
//Imprimos el nombre, la edad y la localidad si tiene
String nombre;
Integer edad;
JSONObject direccion;

for (int i = 0; i < ja_listacontactos.length(); i++) {
    json = (JSONObject)ja_listacontactos.get(i);

    System.out.println("Objeto "+i+":");
    //Sacamos el nombre usando opt. Hay que controlar nulos
    nombre = (String)json.opt("nombre");
    if (nombre!=null) System.out.println("   -nombre:"+nombre);

    //Sacamos la edad usando has y get. Si estamos seguros de que
    //tiene edad, no es necesario el has
    if (json.has("edad")){
        System.out.println("   -edad:"+json.getInt("edad"));
    }

    //Sacamos la localidad usando otra vez has y get
    if (json.has("direccion")){
        direccion = (JSONObject)json.get("direccion");
        if (direccion.has("localidad")){
          System.out.println("
-localidad:"+direccion.getString("localidad"));
        }
    }
    else{
            System.out.println("   -localidad:NO TIENE");
    }

}
```

```
Objeto 0:
   -nombre:Juan
   -edad:19
   -localidad:Lorca
Objeto 1:
   -nombre:Pepe
   -edad:66
   -localidad:NO TIENE
```