Introducción a JSON.

JSON (JavaScript Object Notation) es un formato utilizado en muchos entornos para el intercambio de datos estructurados.

En la web oficial de JSON hay enlaces a librerías para el procesamiento de este formato en numerosos lenguajes, incluyendo PHP, Perl y Java. En el caso de éste último, hay más de veinte enlaces a distintas implementaciones de la funcionalidad de proceso de formato JSON.

En este caso se examinan las posibilidades que ofrece google-gson, la implantación ofrecida por Google, con ejemplos de código java para la lectura y escritura de archivos en formato JSON.

Descarga e instalación de la librería

Acceder a la página principal de google-gson y descargar el archivo google-gson-2.2.4-release.zip (558 KB), o la más actual. Descomprimir el archivo. En su interior, junto con las fuentes y la documentación, se encuentra el archivo que contiene la librería:

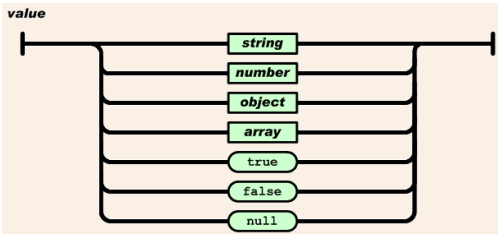
```
google-gson-2.2.4/gson-2.2.4.jar
```

Como con cualquier otra librería java, este archivo debe ser accesible a través de la variable de entorno `CLASSPATH`, o ser incluido explícitamente a la hora de compilar y ejecutar un programa que la utilice.

Formato JSON

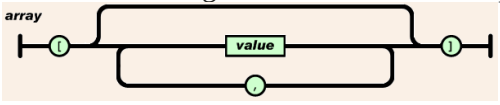
El contenido de un archivo en formato JSON puede ser:

- Un elemento simple (una cadena de texto, un número, un valor booleano, o el valor nulo). Las cadenas de texto se encierran entre comillas, los valores booleanos pueden ser `true` o `false`, y el valor nulo se representa por la palabra clave `null`. Ejemplos:
    - `"Luis Fernandez"`
    - `32`
    - `false`
    - `null`



- Un conjunto ordenado (arreglo), de elementos, separados por comas, y encerrados entre corchetes cuadrados `[` y `]`. Cada uno de los elementos del arreglo puede ser un elemento simple, un arreglo o un conjunto de pares (clave, valor). Ejemplos:
    - `[ "Luis", 32 ]`
    - `[ "Luis", 32, [ "Música", "Cine"], "Madrid"]`

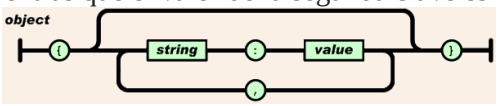
  El primer ejemplo es un arreglo de dos elementos simples, mientras que el segundo ejemplo es un arreglo de cuatro elementos, de los cuales el tercero es a su vez un arreglo de dos elementos simples.



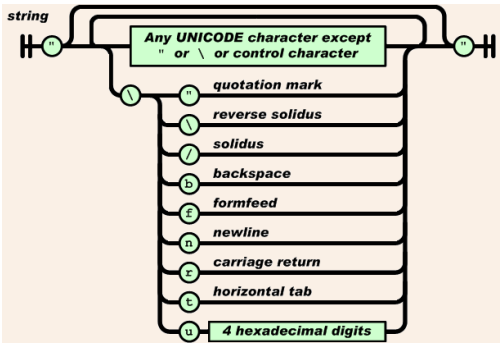
- Un conjunto de pares (clave, valor). Este tipo de estructura también es conocida como hashtable o arreglo asociativo). La clave es una cadena de texto encerrada entre comillas, y el valor es un elemento simple, un arreglo u otro hashtable. La clave se separa del valor por el carácter `:` y el elemento se encierra entre llaves `{` y `}`.
  Ejemplo:

- { "responsable": "Juan", "empleados": ["Elena", "Luis"] }

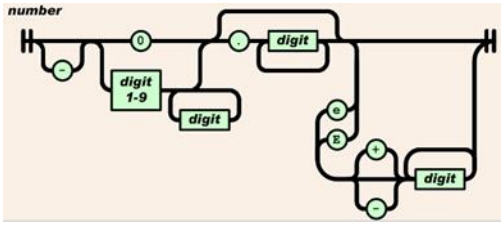
En este ejemplo, el documento es un hashtable con dos claves, "responsable" y "empleados". El valor de la primera clave es un elemento simple, mientras que el valor de la segunda clave es un arreglo.



Un valor puede ser una cadena entre comillas dobles, o un número, o verdadero o falso, o nulo, o un objeto o una matriz. Estas estructuras se pueden anidar.



Una cadena es una secuencia de cero o más caracteres Unicode, envuelta en comillas dobles, con escapes de barra invertida. Un carácter se representa como una sola cadena de caracteres. Una cadena es muy parecido a una de C o una cadena de Java.



El espacio en blanco se puede insertar entre cualquier par de tokens. Exceptuando algunos detalles de codificación, que describe completamente el lenguaje.

Archivo JSON de ejemplo

En este caso, se utiliza el siguiente archivo de ejemplo datos.json:

```
{
"responsable":
    {
    "Nombre" : "Juan",
    "Edad": 28,
    "Aficiones": ["Música", "Cine", "Tenis"],
    "Residencia": "Madrid"
    },
"empleados":
    [
        {
        "Nombre" : "Elena",
        "Edad": 26,
        "Aficiones": ["Música", "Cine"],
        "Residencia": "Madrid"
        },
        {
        "Nombre" : "Luis",
        "Edad": 31,
        "Aficiones": ["Teatro", "Cine", "Fútbol"],
```

```
        "Residencia": "Madrid"
        }
    ]
}
```

Lectura del archivo JSON en Java

Para leer el archivo, se escribe un programa java lee_json, en el que se utiliza la clase `com.google.gson.JsonParser`. Esta clase lee un archivo y devuelve un objeto del tipo `com.google.gsonJsonElement`:

```
import java.io.FileReader;
import com.google.gson.JsonParser;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonArray;
import com.google.gson.JsonPrimitive;
import java.util.Map.Entry;

public class lee_json {
    public static void main(String args[]) throws java.io.IOException {
        JsonParser parser = new JsonParser();
        FileReader fr = new FileReader("datos.json");
        JsonElement datos = parser.parse(fr);
        dumpJSONElement(datos);
    }

    public static void dumpJSONElement(JsonElement elemento) {
        ...
    }
}
```

El método `dumpJSONElement` debe determinar el tipo de elemento que recibe como argumento (elemento simple, hashtable o arreglo), y procesarlo en consecuencia.

Si el elemento recibido es un elemento compuesto de otros elementos (arreglo o hashtable), `dumpJSONElement` se llama a sí mismo recursivamente:

```
public static void dumpJSONElement(JsonElement elemento) {
    if (elemento.isJsonObject()) {
        // Es un conjunto de pares clave, valor
        // Para cada par, imprimir la clave y llamar a dumpJSONElement(valor)
        ...
    } else if (elemento.isJsonArray()) {
        // Es un conjunto de valores, que pueden ser elementos simples o compuestos
        // Para cada valor, llamar a dumpJSONElemento(valor)
        ...
    } else if (elemento.isJsonPrimitive()) {
        // Es un elemento simple. Determinar si se trata de un valor booleano,
        // un número o cadena de texto
        ...
    } else if (elemento.isJsonNull()) {
        System.out.println("Es NULL");
    } else {
        System.out.println("Es otra cosa");
    }
}
```

El código fuente completo de la rutina `dumpJSONElement` es:

```
public static void dumpJSONElement(JsonElement elemento) {
    if (elemento.isJsonObject()) {
```

```
        System.out.println("Es objeto");
        JsonObject obj = elemento.getAsJsonObject();
        java.util.Set<java.util.Map.Entry<String,JsonElement>> entradas = obj.entrySet();
        java.util.Iterator<java.util.Map.Entry<String,JsonElement>> iter =
entradas.iterator();
        while (iter.hasNext()) {
            java.util.Map.Entry<String,JsonElement> entrada = iter.next();
            System.out.println("Clave: " + entrada.getKey());
            System.out.println("Valor:");
            dumpJSONElement(entrada.getValue());
        }

    } else if (elemento.isJsonArray()) {
        JsonArray array = elemento.getAsJsonArray();
        System.out.println("Es array. Numero de elementos: " + array.size());
        java.util.Iterator<JsonElement> iter = array.iterator();
        while (iter.hasNext()) {
            JsonElement entrada = iter.next();
            dumpJSONElement(entrada);
        }
    } else if (elemento.isJsonPrimitive()) {
        System.out.println("Es primitiva");
        JsonPrimitive valor = elemento.getAsJsonPrimitive();
        if (valor.isBoolean()) {
            System.out.println("Es booleano: " + valor.getAsBoolean());
        } else if (valor.isNumber()) {
            System.out.println("Es numero: " + valor.getAsNumber());
        } else if (valor.isString()) {
            System.out.println("Es texto: " + valor.getAsString());
        }
    } else if (elemento.isJsonNull()) {
        System.out.println("Es NULL");
    } else {
        System.out.println("Es otra cosa");
    }
}
```

Una vez escrito el programa lee_json.java, se compila y ejecuta con los comandos:

```
$ javac -cp google-gson-2.2.4/gson-2.2.4.jar lee_json.java
$ java -cp .:google-gson-2.2.4/gson-2.2.4.jar lee_json
```

Y se obtiene el siguiente resultado en pantalla:

```
Es objeto
Clave: responsable
Valor:
Es objeto
Clave: Nombre
Valor:
Es primitiva
Es texto: Juan
Clave: Edad
Valor:
Es primitiva
Es numero: 28
Clave: Aficiones
Valor:
Es array. Numero de elementos: 3
....
```

Escritura de un Archivo JSON

Para convertir un objeto java en una cadena de texto en formato JSON, utilizamos el método `toJson` de la clase `com.google.gson.Gson`.

Si el objeto es de un tipo derivado de la clase `Collection` (`ArrayList`, `Vector`,...), `toJson` lo convierte a un arreglo.

Si el objeto es de otro tipo, `toJson` convierte sus atributos en un conjunto de pares (`clave`, `valor`), en donde `clave` es el nombre del atributo, y `valor` es el valor que le ha sido asignado.

Ejemplos:

1. Tipos simples. El código para convertir a JSON un `String`, `Integer`, etc es el siguiente:

```
import com.google.gson.Gson;
public class escribe_json {
    public static void main(String args[]) throws java.io.IOException {
        Gson gson = new Gson();
        String datos = "Hola";
        String jsonString = gson.toJson(datos);
        System.out.println("JSON: " + jsonString);
    }
}
```

El código del ejemplo escribe en pantalla el siguiente resultado:

```
JSON: "Hola"
```

2. Colecciones

Si el objeto que se pasa al método `toJson` es de la clase `Collection` o derivada:

```
public static void main(String args[]) throws java.io.IOException {
    Gson gson = new Gson();
    Collection collection = new ArrayList();
    collection.add("hello");
    collection.add(5);
    String json = gson.toJson(collection);
    System.out.println("JSON: " + json);
}
```

Se obtiene como resultado un arreglo:

```
JSON: ["hello",5]
```

3. Objetos

Por último, se puede pasar a `toJson` un objeto. Se define primero una clase `MiObjeto` con atributos simples `nombre`, `origen`, `cadena` y atributos compuestos `miColeccion`, `miVector`:

```
static class MiObjeto {
    private String nombre;
    private String origen;
    String miCadena;
    Collection miColeccion = new ArrayList();
    Vector miVector;
    private MiObjeto(String nombre, String origen, String cadena) {
        this.nombre = nombre;
        this.origen = origen;
        this.miCadena = cadena;
        miColeccion.add("adios");
        miColeccion.add(10);
        miVector = new Vector();
        miVector.add("Elemento1");
        miVector.add(null);
```

```
        miVector.add("Elemento3");
        miVector.add("Elemento4");
    }
}
```

A continuación, en el programa principal, se crea una instancia del objeto y se convierte a un string `JSON`:

```
MiObjeto obj = new MiObjeto("Juan", "Madrid", null);
String jsonString = gson.toJson(obj);
System.out.println("JSON: " + jsonString);
```

y se obtiene como resultado:

```
JSON: {"nombre":"Juan","origen":"Madrid","miColeccion":["adios",10],
       "miVector":["Elemento1",null,"Elemento3","Elemento4"]}
```

Por último, `toJson` ha convertido el objeto en un conjunto de pares (`clave`, `valor`), en donde las claves son los nombres de los atributos. Los valores compuestos de tipo `Collection` (`miColeccion`, `miVector`) han sido transformados en arreglos. El valor nulo asignado a uno de los elementos de `miVector` aparece en el arreglo y resultante. El atributo `miCadena`, al que se le ha asignado un valor nulo, no aparece en el string `JSON` resultante.

Nota: Generar un reporte del ejercicio. Cargar en la red la carpeta de archivos y el reporte, con nombre de alumno y tipo de ejercicio.