Gson

Open-source Java library to serialize and deserialize Java objects to and from JSON

Google Gson is a simple Java-based library to serialize Java objects to JSON and vice versa. It is an open-source library developed by Google.

The following points highlight why you should be using this library −

* **Standardized** − Gson is a standardized library that is managed by Google.
* **Efficient** − It is a reliable, fast, and efficient extension to the Java standard library.
* **Optimized** − The library is highly optimized.
* **Support Generics** − It provides extensive support for generics.
* **Supports complex inner classes** − It supports complex objects with deep inheritance hierarchies.

Features of Gson

Here is a list of some of the most prominent features of Gson −

* **Easy to use** − Gson API provides a high-level facade to simplify commonly used use-cases.
* **No need to create mapping** − Gson API provides default mapping for most of the objects to be serialized.
* **Performance** − Gson is quite fast and is of low memory footprint. It is suitable for large object graphs or systems.
* **Clean JSON** − Gson creates a clean and compact JSON result which is easy to read.
* **No Dependency** − Gson library does not require any other library apart from JDK.
* **Open Source** − Gson library is open source; it is freely available.

## Three Ways of Processing JSON

Gson provides three alternative ways to process JSON −

### **Streaming API**

It reads and writes JSON content as discrete events. **JsonReader** and **JsonWriter** read/write the data as token, referred as **JsonToken**.

It is the most powerful approach among the three approaches to process JSON. It has the lowest overhead and it is quite fast in read/write operations. It is analogous to Stax parser for XML.

### **Tree Model**

It prepares an in-memory tree representation of the JSON document. It builds a tree of JsonObject nodes. It is a flexible approach and is analogous to DOM parser for XML.

### **Data Binding**

It converts JSON to and from POJO (Plain Old Java Object) using property accessor. Gson reads/writes JSON using data type adapters. It is analogous to JAXB parser for XML.

What is JSON and why we use it?

* - It has no tag format. - It is storage optimal as it is relatively shorter than XML. - It is faster to read and write. - It can use arrays while it is not easy in XML.

What is JSON?

HTML

JSON

JSON stands for JavaScript Object Notation

JSON is a lightweight format for storing and transporting data

JSON is often used when data is sent from a server to a web page

JSON is "self-describing" and easy to understand

JSON Example

This example defines an employees object: an array of 3 employee records (objects):

{

"employees":[

{"firstName":"John", "lastName":"Doe"},

{"firstName":"Anna", "lastName":"Smith"},

{"firstName":"Peter", "lastName":"Jones"}

]

}

JSON Syntax Rules

Data is in name/value pairs

Data is separated by commas

Curly braces hold objects

Square brackets hold arrays

JavaScript Object Notation

The JSON format is syntactically identical to the code for creating JavaScript objects.

Because of this similarity, a JavaScript program can easily convert JSON data into native JavaScript objects.

The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

JSON Data - A Name and a Value

JSON data is written as name/value pairs, just like JavaScript object properties.

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

"firstName":"John"

JSON names require double quotes. JavaScript names do not.

JSON Objects

JSON objects are written inside curly braces.

Just like in JavaScript, objects can contain multiple name/value pairs:

{"firstName":"John", "lastName":"Doe"}

JSON Arrays

JSON arrays are written inside square brackets.

Just like in JavaScript, an array can contain objects:

"employees":[

{"firstName":"John", "lastName":"Doe"},

{"firstName":"Anna", "lastName":"Smith"},

{"firstName":"Peter", "lastName":"Jones"}

]

In the example above, the object "employees" is an array. It contains three objects.

Each object is a record of a person (with a first name and a last name).

Converting a JSON Text to a JavaScript Object

A common use of JSON is to read data from a web server, and display the data in a web page.

For simplicity, this can be demonstrated using a string as input.

First, create a JavaScript string containing JSON syntax:

var text = '{ "employees" : [' +

'{ "firstName":"John" , "lastName":"Doe" },' +

'{ "firstName":"Anna" , "lastName":"Smith" },' +

'{ "firstName":"Peter" , "lastName":"Jones" } ]}';

Then, use the JavaScript built-in function JSON.parse() to convert the string into a JavaScript object:

var obj = JSON.parse(text);

Finally, use the new JavaScript object in your page:

Example

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML =

obj.employees[1].firstName + " " + obj.employees[1].lastName;

</script>

# **Jackson Tutorial**

**Jackson** is a high-performance JSON processor used for Java. It is the most popular library used for serializing Java objects or Map to JSON and vice-versa. It is completely based on Java. Jackson tutorial provides all the basic and advanced concepts of the Jackson library.

JSON is one of the most important data interchange formats that is mainly used in the world of Web applications. JSON requests can be easily parsed by the browser can be easily converted into JavaScript objects but cannot be easily converted into Java objects.

Serialization and deserialization are two important concepts when working with JSON. Serialization is a process of converting Java objects into JSON whereas deserialization is a process of converting JSON into Java objects.

* We need a parser for parsing Json or converting them into Java Objects. We parse JSON into Object because it is not that easy to work with JSON strings.
* Jackson is a Java Json library that has a built-in ObjectMapper class. The ObjectMapper class is responsible for parsing the JSON files and deserializing them into Java objects.
* It provides JSON Parser and JSON Generator that helps us in parsing and generating json one token at a time.

Features of Jackson library

Jackson library has the following features:

**1. Easy to use**

In order to simplify the common use cases, Jackson provides a high-level façade.

**2. Performance**

Jackson library is quite fast and has a low memory footprint. It is suitable for large systems.

**3. No Dependency**

In order to use Jackson, there is no need to use any other library except JDK. JDK is required because Jackson is based on Java.

**4. Mapping free**

There is no need to create mapping because it is by default provided for most of the objects to be serialized.

**5. Open Source**

It is freely available in the market. There is no need to purchase anything to use Jackson library. Just download it and use it in the code.

**6. Clean JSON**

It converts an object or a Map into a clean and compact JSON that is very easy to read.

Processing JSON

In order to process JSON, Jackson provides three ways which are as follows:

**1. Streaming API**

By using streaming API, JSON data read and write as discrete events. Streaming API provides JsonParser and JsonGenerator for reading and writing data, respectively. Streaming API is one of the fastest and lowest overheads in read/write operations. It is the most powerful approach among others.

**2. Tree Model**

For reading and writing JSON data, it creates a tree representation of JSON documents. The ObjectMapper class is used to create a tree of nodes (JsonNode). However, it is not as faster as streaming API, but it is the most flexible approach for reading and writing JSON data.

**3. Data Binding**

It is a way to convert POJO to JSON and JSON to POJO. It is done by using annotations or a property accessor. Data binding is of two types, i.e., Simple Data Binding and Full Data Binding.

Simple Data Binding is used to convert JSON to and from null Object, Map, List, String, Number, and Boolean.

Full Data Binding is used to convert JSON to and from any of the Java types.

Prerequisite

Jackson is completely based on Java, so before proceeding with this tutorial, it is required that you should have knowledge of Java.