gSON

java library

tutorialspoint

SIMPLY EASY LEARNING

About the Tutorial

Google Gson is an open source, Java-based library developed by Google. It facilitates serialization of Java objects to JSON and vice versa.

This tutorial adopts a simple and intuitive way to describe the basic-to-advanced concepts of Google Gson and how to use its APIs.

Audience

This tutorial will be useful for most Java developers, starting form beginners to experts. After completing this tutorial, we are confident that you will find it easy to use Google Gson in your programs.

Prerequisites

It is a simple tutorial that any developer with a little exposure to Java programming can easily understand.

Copyright & Disclaimer

Table of Contents

1. GSON – OVERVIEW

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.

2. GSON – ENVIRONMENT SETUP

Try it Option Online

You do not have to set up your own environment to start learning Gson. We have already set up an online Java Programming environment for you. This platform allows you to compile and execute all the available examples online and analyze the output with different options. We believe it will boost your confidence and make your learning an enjoyable activity. Feel free to modify any example and execute it online.

Try the following example using the **Try it** option available at the top right corner of the following sample code box:

```
public class MyFirstJavaProgram {


    public static void main(String []args) {

       System.out.println("Hello World");

    }

}
```

For most of the examples given in this tutorial, you will find a **Try it** option. Please do use this option and enjoy your learning.

Local Environment Setup

If you still want to set up a local environment for Java programming language, then this section will guide you on how to download and set up Java on your machine. Please follow the steps given below, to set up the environment.

Java SE is freely available from the link Download Java. You need to download a version based on your operating system.

Follow the instructions to download Java and run the **.exe** to install Java on your machine. Once you have installed Java on your machine, you would need to set the environment variables to point to their correct installation directories.

Setting up the Path in Windows 2000/XP

Assuming you have installed Java in *c:\Program Files\java\jdk* directory:

- Right-click on 'My Computer' and select 'Properties'.

- Click on the 'Environment variables' button under the 'Advanced' tab.

- Next, alter the 'Path' variable so that it also contains the path to the Java executable. For example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;c:\Program Files\java\jdk\bin'.

Setting up the Path in Windows 95 / 98 / ME

Assuming you have installed Java in *c:\Program Files\java\jdk* directory:

- Edit the 'C:\autoexec.bat' file and add the following line at the end: 'SET PATH=%PATH%;C:\Program Files\java\jdk\bin'

Setting up the Path for Linux, UNIX, Solaris, FreeBSD

The environment variable **PATH** should be set to point to where the Java binaries have been installed. Refer to your shell documentation if you have trouble doing this.

For example, if you use *bash* as your shell, then you would add the following line to the end of your '.bashrc: export PATH=/path/to/java:$PATH'

Popular Java Editors

To write your Java programs, you will need a text editor. There are quite a few sophisticated IDEs available in the market. But for now, you can consider one of the following:

- **Notepad:** On Windows, you can use any simple text editor like Notepad (Recommended for this tutorial) or TextPad.

- **Netbeans:** It is a Java IDE that is open-source and free which can be downloaded from http://www.netbeans.org/index.html.

- **Eclipse:** It is also a Java IDE developed by the Eclipse open-source community and can be downloaded from http://www.eclipse.org/.

Download Gson Archive

Download the latest version of Gson jar file from gson-2.3.1.jar. At the time of writing this tutorial, we downloaded *gson-2.3.1.jar* and copied it into C:\>gson folder.

OS	Archive name
Windows	gson-2.3.1.jar
Linux	gson-2.3.1.jar
Mac	gson-2.3.1.jar

Set Gson Environment

Set the **GSON_HOME** environment variable to point to the base directory location where Gson jar is stored on your machine.

OS	Output
Windows	Set the environment variable GSON_HOME to C:\gson
Linux	export GSON_HOME=/usr/local/gson
Mac	export GSON_HOME=/Library/gson

Set CLASSPATH variable

Set the **CLASSPATH** environment variable to point to the Gson jar location.

OS	Output
Windows	Set the environment variable CLASSPATH to %CLASSPATH%;%GSON_HOME%\gson-2.3.1.jar;.;
Linux	export CLASSPATH=$CLASSPATH:$GSON_HOME/gson-2.3.1.jar:.
Mac	export CLASSPATH=$CLASSPATH:$GSON_HOME/gson-2.3.1.jar:.

3. GSON – FIRST APPLICATION

Before going into the details of the Google Gson library, let's see an application in action. In this example, we've created a **Student** class. We'll create a JSON string with student details and deserialize it to **student** object and then serialize it to an JSON String.

Example

Create a Java class file named GsonTester in C:\>GSON_WORKSPACE.

File: GsonTester.java

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

public class GsonTester {
   public static void main(String[] args){
      String jsonString = "{\"name\":\"Mahesh\", \"age\":21}";
      GsonBuilder builder = new GsonBuilder();
      builder.setPrettyPrinting();
      Gson gson = builder.create();
      Student student = gson.fromJson(jsonString, Student.class);
      System.out.println(student);
      jsonString = gson.toJson(student);
      System.out.println(jsonString);
   }
}

class Student {
   private String name;
   private int age;
   public Student(){}
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public int getAge() {
      return age;
   }
   public void setAge(int age) {
      this.age = age;
   }
   public String toString(){
      return "Student [ name: "+name+", age: "+ age+ " ]";
   }
}
```

Verify the result

Compile the classes using **javac** compiler as follows:

```
C:\GSON_WORKSPACE>javac GsonTester.java
```

Now run the GsonTester to see the result:

```
C:\GSON_WORKSPACE>java GsonTester
```

Verify the output.

```
Student [ name: Mahesh, age: 21 ]
{
  "name" : "Mahesh",
  "age" : 21
}
```

Steps to Remember

Following are the important steps to be considered here.

Step 1: Create Gson object using GsonBuilder

Create a Gson object. It is a reusable object.

```
GsonBuilder builder = new GsonBuilder();

builder.setPrettyPrinting();

Gson gson = builder.create();
```

Step 2: Deserialize JSON to Object

Use fromJson() method to get the Object from the JSON. Pass Json string / source of Json string and object type as parameter.

```
//Object to JSON Conversion

Student student = gson.fromJson(jsonString, Student.class);
```

Step 3: Serialize Object to JSON

Use toJson() method to get the JSON string representation of an object.

```
//Object to JSON Conversion

jsonString = gson.toJson(student);
```