Jacob Cannamela

CSD 420

5/3/2025

**Module 11 Assignment**

One of the most popular API’s for working with JSON in Java is Google's Gson library. The integration of Gson into Java applications is so clean and simple that it’s widely adopted. As a novice working with JSON data, it was easy for me to work with Gson because it doesn’t require a lot of unnecessary code. Its basic functions of converting Java objects into JSON and vice versa really enable developers to focus on enhancing their applications rather than on complexity.

Google developed Gson and released it back in 2008. It became part of the Google ecosystem due to the growing need of efficiently serializing and deserializing Java objects for service communication, especially during Android development. The value of its simple to use api ensures it keeps being maintained on GitHub. Unlike other older libraries that depend on annotations and extra configurations, Gson’s ability of performing conversion using plain Java objects makes it a great choice for developers who want a lean implementation.

Gson’s complex or generic data types can be processed with something referred to as TypeToken. This feature allows preserving type data and deserialization, further enhancing list, map, or any customized collection manipulations. Gson also supports and handles nested objects, null values, and customizable serializers and deserializers. This means that when a user has specific guidelines on how to format a Java object to JSON, they can write their own rules, and where the logic is applicable, Gson can follow. When trying to format date and time values, or exclude certain fields from the output, I found this very useful for the task at hand.

Another major benefit is how light Gson is. Because it doesn't have a lot of dependencies, you won’t bloat your projects just to include JSON functionality. It is as simple as including the JAR file and utilizing it with almost no configuration. The library supports streaming and tree structure approaches for parsing JSON. The streaming API is particularly helpful for reading big files, as it does so in chunks, minimizing memory consumption. On the other hand, the tree model is useful for dynamic parsing where the exact shape is unknowable in advance.

Developers working on a project can choose to download the latest Gson JAR from either the official Maven repository or [GitHub](https://github.com/google/gson). The download comes with all files essential to add Gson to their Java projects. For students or workgroups that prefer having everything in one place, the URL https://search.maven.org/artifact/com.google.code.gson/gson/2.13.1/jar?eh= provides a zip file containing the jar file.

In conclusion, Gson is the best Java API for JSON, it is both flexible and developer friendly due to being so fast and simple yet having so many features for advanced JSON manipulation. As someone just starting in software development, I especially appreciated the ease of practical work like parsing API responses and constructing configuration files. Its popularity in desktop applications as well as Android speaks for itself.

**References:**

*Gson - Quick Guide*. (n.d.). Www.tutorialspoint.com. https://www.tutorialspoint.com/gson/gson\_quick\_guide.htm

‌ GeeksforGeeks. (2022, February 27). *How to Install GSON Module in Java?* GeeksforGeeks. https://www.geeksforgeeks.org/how-to-install-gson-module-in-java/

‌ Therrien, A. (2020, June 2). *Building a Personalized Serializer and Deserializer using Java Gson Library*. Medium. https://medium.com/@alexandre.therrien3/personalized-serializer-and-deserializer-using-java-gson-library-c079de3974d4

‌ Wikipedia Contributors. (2024, July 29). *Gson*. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Gson#:~:text=6%20Further%20reading-,History,of%20the%20Apache%20License%202.0.

‌