JSON, or JavaScript Object Notation, is a way to format data that allows developers to easily read and understand it. There are many processors and open-source tools for processing JSON in Java, but a popular choice among developers is Jackson because of its ease to use, flexibility, and simplicity. The FasterXML team that maintains Jackson through open-source files on Github provides tools for parsing, generating, transforming, and querying JSON data in a variety of Java applications.

Over the years since Jackson was created, Jackson has grown to include many different sub-series that are focused on specific purposes, and integration methods to allow use across different data formats. The different specialized modules allow developers to customize their use, allowing programs to stay lightweight and only use what they need; the expanded functions also help shorten the time developers spend on specific aspects of their projects. JSON's whole purpose is to give developers a simple and readable data format, and the large number of resources within Jackson helps give developers a wide range of access to open-source tools while keeping the elements lightweight and customizable for each Java project it's used for.

As stated above, the biggest advantage to using Jackson for processing JSON projects is the flexibility it allows the developer, while also keeping the projects lightweight. There are many other features of Jackson that make it as popular among developers as it is. Jackson smoothly integrates with many frameworks, which is why it’s the default mapper in many Java web applications. Projects with large data volume are perfect for Jackson since it allows reading and writing JSON to be done incrementally. The flexibility also allows developers to define Java types are serialized or deserialized and allows them to determine how specific they want to them to be. It also allows developers to customize JSON mapping.

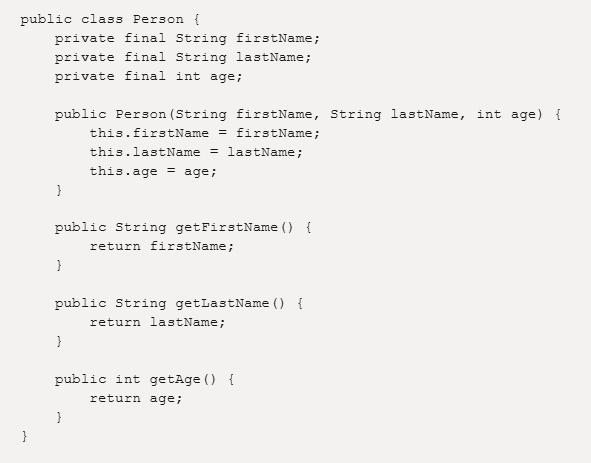
Jackson is used more often for large, data-heavy projects because it’s the API with the best performance. Google also has a similar API called “Gson”, and while it has a lot of similar features, it’s mostly used for object mapping. It’s not as customizable, and is often slower for larger datasets, making it okay for smaller projects but not for anything with large datasets. Gson is very simple to understand, and when compared, Jackson has a larger learning curve. But the many features and advantages of Jackson make the effort to learn worth it if you plan on using projects with large datasets.

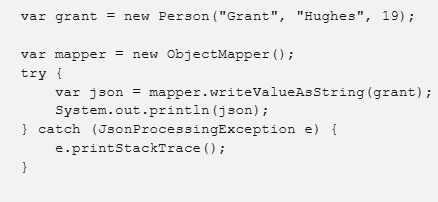
Jackson is a large library that can support both large and small Java applications. It’s flexible and customizable, and is the fastest JSON API available, and is the most popular API used for applications with large data sets. Developers can choose what they want to use in their projects, making it more lightweight. It seamlessly integrates with frameworks and helps format data within applications to be easily read and understood by developers. Jackson is easily found and accessed through Github, and the FasterXML team maintains the library, and helps to make it easily accessible and safe to use for developers.

Where to access:

[FasterXML/jackson: Main Portal page for the Jackson project](https://github.com/FasterXML/jackson)

Usage example:





Resources:

[Jackson (API) - Wikipedia](https://en.wikipedia.org/wiki/Jackson_(API)#:~:text=In%20computing%2C%20Jackson%20is%20a,ergonomic%20attributes%20of%20the%20library.)

[FasterXML/jackson: Main Portal page for the Jackson project](https://github.com/FasterXML/jackson)

[Jackson JSON Java Parser API Example Tutorial | DigitalOcean](https://www.digitalocean.com/community/tutorials/jackson-json-java-parser-api-example-tutorial)

[Jackson JSON Series | Baeldung](https://www.baeldung.com/jackson)

[Efficient JSON serialization with Jackson and Java](https://blogs.oracle.com/javamagazine/post/java-json-serialization-jackson)