JACKSON JSON API

Name: Christopher Reaney
Assignment: Module 11-2 Assignment
Date: 05/09/2025

JSON, (JavaScript Object Notation) is a common way for programs to share data. It's used all the time in web development and apps, especially when sending information between a server and a client. Java doesn't have built-in tools for working with JSON, so most Java developers use a third-party library to handle it. One of the best-known libraries for this is Jackson. Jackson is fast, flexible, and pretty easy to use once you get the hang of it. This paper will take a closer look at Jackson; What it can do, how it works, a bit about its background, and where to get it if you want to use it in your own projects.

Jackson was created in 2009 by Tatu Saloranta, who was trying to make something faster and more flexible than other JSON tools for being used in Java at the time (FasterXML, 2023). Over time, Jackson became one of the most popular libraries for handling JSON in Java, especially after being included as the default in Spring Boot: a widely used framework. It's now maintained by the open-source team at FasterXML and receives relatively regular updates.

Jackson is made up of a few main components. The first would be jackson-core, which is for low-level reading and writing of JSON data. After which there is jackson-databind, which is what most people use. It makes it easy to convert Java objects to JSON and back. The third main part is jackson-annotations, which lets you control how your JSON is processed using annotations like @JsonIgnore or @JsonProperty (Baeldung, 2024).

Jackson gives developers different ways to work with JSON. One way is called the streaming API, where you process JSON one token at a time. That can be useful for big files or when memory is limited. Another option is the tree model, where you work with JSON as if it were a tree of nodes. The most popular and in my opinion also the easiest method is data binding, which uses a class called ObjectMapper. With this approach, you can turn a Java object into a JSON string or vice-versa with just a single line of code. For example, mapper.writeValueAsString(myObject) turns an object into a string, and mapper.readValue(json, MyClass.class) reads it back.

Jackson is also great when you need to customize how your JSON is processed. You can write your own serializers or deserializers, work with generic types, handle polymorphism (like abstract classes), and more. It also integrates nicely with Java 8 features like Optionals and Streams, as well as Java time classes like LocalDate.

If you want to use Jackson in your own Java project, you can find the source code and documentation on GitHub at https://github.com/FasterXML/jackson. The JAR files can be downloaded from Maven Central, where all the necessary modules like jackson-core, jackson-databind, and jackson-annotations are available. You can download them separately or in a zip file if you're not using a build tool like Maven or Gradle (Maven Central Repository, 2025). For Gradle, adding it in should be as easy as including the library and adding dependencies in the build.gradle file.

Overall, Jackson is a solid choice for working with JSON in Java. It’s fast, well-supported, and easy to learn. Whether you’re building something small for a class project or working on a big web service, Jackson has the tools to help you manage your data efficiently.

References

Baeldung. (2024). Intro to the Jackson ObjectMapper. Retrieved from https://www.baeldung.com/jackson-object-mapper-tutorial

FasterXML. (2023). Jackson JSON Processor. GitHub. Retrieved from https://github.com/FasterXML/jackson

Maven Central Repository. (2025). com.fasterxml.jackson.core. Retrieved from https://search.maven.org/artifact/com.fasterxml.jackson.core/jackson-core