**Amanda Wedergren**

**October 6, 2025**

**Module 11.2 Assignment**

**Exploring Jackson: A Powerful Java JSON API for Modern Development**

When working with JSON in Java, one of the most reliable and widely adopted libraries is Jackson. It’s often referred to as “the Java JSON library” because of its speed, flexibility, and ease of use. Whether you're serializing Java objects to JSON or deserializing JSON into Java objects, Jackson offers a robust set of tools to get the job done efficiently. I chose Jackson for this paper because it’s not only popular in the Java community, but also incredibly practical for real-world applications.

Jackson was originally developed by Tatu Saloranta in 2007 and has since evolved into a comprehensive suite of data-processing tools for Java. The library gained popularity due to its performance and modular design. Over time, it expanded beyond basic JSON parsing to support other data formats like XML, YAML, CSV, and even binary formats like Smile and CBOR. This versatility makes it a great choice for developers who need to work with multiple data formats without switching libraries.

One of the things I appreciate most about Jackson is its modular architecture. It’s composed of three core modules: jackson-core, jackson-databind, and jackson-annotations. The jackson-core module provides the low-level streaming API for reading and writing JSON. This is especially useful when working with large datasets, as it allows for efficient parsing without loading the entire document into memory. The jackson-databind module is the high-level data-binding API that allows conversion between Java objects and JSON. Finally, jackson-annotations supports annotations used to customize serialization and deserialization behavior.

Jackson supports several processing models, which gives developers flexibility depending on their needs. The streaming model uses JsonParser and JsonGenerator for token-based reading and writing. The tree model uses JsonNode to represent JSON as a tree structure, which is helpful when dealing with dynamic or unknown schemas. The data-binding model, which is the most commonly used, allows developers to map JSON directly to Java objects using the ObjectMapper class. This class is essentially the workhorse of Jackson and makes it incredibly easy to serialize and deserialize data with just a few lines of code.

Another standout feature is Jackson’s support for annotations. Developers can use annotations like @JsonProperty, @JsonIgnore, and @JsonInclude to control how fields are serialized or deserialized. This makes Jackson highly adaptable to different data structures and use cases. For example, if you only want to include non-null fields in your JSON output, you can simply use @JsonInclude(JsonInclude.Include.NON\_NULL).

Jackson also integrates well with other frameworks, especially Spring Boot. In fact, Spring Boot uses Jackson as its default JSON processor, which speaks volumes about its reliability and performance. It’s also worth noting that Jackson is actively maintained, with a strong community and frequent updates that keep it compatible with the latest versions of Java.

If you’re not using a build tool like Maven or Gradle, you can manually download the Jackson JAR files. The three essential JARs—jackson-core, jackson-databind, and jackson-annotations—can be found on the Maven Repository. For convenience, you can download all JARs zipped together from the GitHub repository or use a Maven BOM to ensure version compatibility.

In conclusion, Jackson is a powerful and flexible JSON API that offers high performance, modular design, and extensive customization options. Its ability to handle multiple data formats and processing models makes it a top choice for Java developers. Whether you're building RESTful APIs, integrating with third-party services, or managing complex data structures, Jackson provides the tools to do it efficiently and effectively. I highly recommend exploring Jackson if you’re looking for a reliable JSON solution in Java.

References

FasterXML. (2025). *Jackson Project Portal*. GitHub. https://github.com/FasterXML/jackson

Javarevisited. (2017, June). *How to get JAR files of Jackson Library for JSON Processing*. https://javarevisited.blogspot.com/2017/06/how-to-get-jar-files-of-jackson-libary-for-JSON.html

Stack Overflow. (n.d.). *Downloading jackson.codehaus.org jar*. https://stackoverflow.com/questions/12539270/downloading-jackson-codehaus-org-jar