Lea Trueworthy

May 18, 2025

M11: Programming Assignment

**Jackson JSON API: A Comprehensive Overview**

The Jackson JSON API is one of the most popular libraries used in Java for parsing, generating, and manipulating JSON data. Known for its high performance, flexibility, and ease of use, Jackson provides a set of tools that make working with JSON straightforward. The Jackson library is primarily used for two main purposes: converting Java objects to JSON (serialization) and converting JSON back into Java objects (deserialization).

**History and Evolution of Jackson**

Jackson began as a small project intended to provide lightweight JSON support for Java, but over time it evolved into a modular and extensible framework. The core Jackson library is accompanied by multiple modules such as jackson-core, jackson-databind, and jackson-annotations, each serving a different purpose in the JSON processing pipeline. Jackson is now maintained by FasterXML and has become a standard library used by frameworks such as Spring Boot for automatic JSON serialization and deserialization.

**Core Components**

Here are the commonly used modules:

* Jackson Core: The jackson-core module provides the foundational functionality for JSON parsing and generation. It is the low-level component responsible for reading and writing JSON, and it's highly optimized for performance.
* Jackson Databind: This module builds on the core and provides data binding capabilities that allow Java objects to be seamlessly converted to and from JSON. It is one of the most commonly used modules in Jackson.
* Jackson Annotations: The jackson-annotations module provides annotations that help control the way objects are serialized and deserialized. These annotations allow users to specify custom behavior for JSON conversion.

**Downloading Jackson JAR Files**

The JAR files for Jackson can be downloaded from the official Maven repository or the GitHub releases page. A zipped bundle of all core Jackson JARs is available at:

<https://github.com/FasterXML/jackson>

To download the JARs directly:

* jackson-core: <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/jackson-core/2.19.0/>
* jackson-annotations: <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/jackson-annotations/3.0-rc4/>
* jackson-databind: <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/jackson-databind/2.19.0/>

**Features and Processes Supported**

Jackson offers many features that make it a powerful tool for JSON processing in Java:

1. High Performance: Jackson is known for its speed in both parsing and generating JSON. This is a significant advantage when working with large datasets or requiring real-time JSON processing (Kumar, 2023).
2. Data Binding: Jackson simplifies the process of converting Java objects to JSON and vice versa. Using ObjectMapper, Jackson automatically maps JSON properties to Java fields, making serialization and deserialization nearly automatic.
3. Annotations for Customization: Jackson provides a rich set of annotations that allow developers to customize how Java objects are serialized or deserialized. For instance, JsonProperty can be used to specify the name of a JSON property, while JsonIgnore can be used to exclude a field from serialization.
4. Streaming API: Jackson's streaming API allows for processing large JSON documents in a memory-efficient manner. This is particularly useful when dealing with large files or real-time data streams.
5. Support for Collections: Jackson handles Java collections (e.g., Lists, Maps) and arrays seamlessly, making it easy to work with complex data structures.
6. Extensibility: Jackson is highly customizable, allowing developers to write custom serializers and deserializers to handle complex data types.

**Example Java**

The example, saved as JacksonExample.java, defines a simple Person class with two fields: name and age. Using Jackson’s ObjectMapper, the program first converts a Person object into a JSON string. Then, it takes that JSON string and converts it back into a Java object, demonstrating both directions of the process.

With just a few lines of code, this approach demonstrates how minimal the setup can be when using Jackson in a standalone application. It also illustrates how the core functionality works without relying on additional frameworks.

References:

Fadatare, R. (2025, March 10). *Jackson Library in Java - Ramesh Fadatare - Medium*. Medium. https://rameshfadatare.medium.com/jackson-library-in-java-eef28ab9fb40

JT. (2017, November 29). *Mastering Jackson Annotations: A Guide to Simplified JSON Mapping for Java -*. Spring Framework Guru - Become a Spring Framework Guru. https://springframework.guru/jackson-annotations-json/

Kumar, P. (2022, August 3). *Jackson JSON Java Parser API Example Tutorial | DigitalOcean*. Www.digitalocean.com. https://www.digitalocean.com/community/tutorials/jackson-json-java-parser-api-example-tutorial

Kumar, R. (2023, August 13). *Jackson JSON - Introduction - Javacodepoint*. Javacodepoint. https://javacodepoint.com/jackson-json-introduction/

Paraschiv, E. (2015, June 7). *Baeldung*. Baeldung. https://www.baeldung.com/jackson-annotations