Jose Franco

05/17/2025

M11 Programming Assignment

Java JSON API

**Java JSON API Jackson**

When working with Java and JSON, one library that really stands out is Jackson. It's been around for a while and has become a go-to solution for developers who want to turn Java objects into JSON and vice versa. Jackson is reliable, fast, and supports different ways of working with JSON, depending on what your project needs.

**Short history of Jackson**

Jackson was created back in 2007 by a developer named Tatu Saloranta. The goal was to make a library that could handle JSON in a fast and flexible way. Since then, it's grown a lot and is now maintained by a team at FasterXML. Jackson has become one of the most widely used Java libraries for handling JSON.

**Features and Capabilities**

Jackson is composed of several core modules that work together or independently depending on a developer’s needs. The three most significant modules are:

1. **jackson-core**: this part handles reading and writing JSON at a very low level.
2. **jackson-databind**: this is the part that lets you map JSON to Java objects and back using a class called ‘ObjectMapper’.
3. **jackson-annotations**: these are special markers you can add to your Java code to control how it turns into JSON.

**Key features**

* **Data binding (ObjectMapper)**: Jackson allows developers to map JSON to Java objects using the ObjectMapper class. This includes support for both simple and complex POJOs (Plain Old Java Objects).
* **Streaming API (JsonParser / JsonGenerator)**: enables developers to read and write JSON in a tokenized format, useful for large data sets or constrained environments.
* **Tree model (JsonNode)**: provides a flexible, in-memory representation of JSON objects similar to DOM trees in XML processing.
* **Annotations**: offers fine-tuned control over serialization and deserialization using Java annotations.
* **Extensibility**: supports modules for XML, YAML, Protobuf, and other data formats through additional add-ons.
* **Integration**: easily integrates with other frameworks like Spring Boot, Jersey, and Apache Camel.

**Typical Processes Supported**

Jackson supports the following primary processes:

* **Serialization**: Converting Java objects into JSON format.
* **Deserialization**: parsing JSON into Java objects.
* **Streaming parsing**: efficiently processing large JSON files by reading them token by token.
* **Custom serialization/deserialization**: developers can define custom behaviors for JSON mapping.
* **Tree Manipulation**: traverse or modify JSON trees using JsonNode and ObjectNode.

**Download Information**

Jackson is open-source and free to use. You can grab the files from GitHub or from Maven Central. If you're not using Maven or Gradle, you can download all the JAR files zipped together:

* **Download page**: <https://github.com/FasterXML/jackson>
* **Central repository**: <https://search.maven.org/search?q=jackson>
* **Direct ZIP (All JARs)**: <https://repo1.maven.org/maven2/com/fasterxml/jackson/core/>

In summary, Jackson is a strong and flexible option for handling JSON in Java. It supports multiple approaches like streaming, data binding, and tree models, giving developers choices based on the project’s needs. Its use of annotations makes customization straightforward, and it integrates well with many frameworks. Because it’s widely trusted and actively maintained, Jackson remains one of the best tools available for managing JSON in Java applications.

“Jackson aims to be the best possible combination of fast, correct, lightweight, and ergonomic for developers” (Saloranta, 2023).

**References**

* Saloranta, T. (2023). *FasterXML Jackson*. GitHub. <https://github.com/FasterXML/jackson>
* Maven Central Repository. (n.d.). *Search Results for Jackson*. <https://search.maven.org/search?q=jackson>
* Baeldung. (2022). Jackson ObjectMapper Tutorial. <https://www.baeldung.com/jackson-object-mapper-tutorial>