Ean Masoner

Module 11 Assignment

CSD-420

05/13/2025

**Java API for JSON Processing (JSR353)**

**Introduction**

JSON (JavaScript Object Notation) is a largely used data exchange format that is very lightweight and simple to read/write. Java provides multiple APIs for working with JSON, but one of the most standard and widely used is the Java API for JSON Processing or JSR 353. This API has both object model and streaming APIs, letting programmers work with JSON efficiently.

**Features and Processes Supported**

1. **Object Model API**

* Works like the Document Object Model for XML
* Allows for random access to JSON data.
* Uses JsonObject and JsonArray to represent JSON structures.
* Good for applications that need manipulation of JSON data.

1. **Streaming API**

* Works like SAX parsing for XML
* Has an event-driven approach to JSON processing
* Uses the JsonParser function for reading JSON and the JsonGenerator for writing it.
* It is more memory efficient than the Object model API.

**Additional Features**

* Standardized: Works across many Java implementations.
* Can be used with Java EE
* Enables the modification of JSON documents.

**History of JSR 353**

JSR 353 was introduced as a part of Java EE 7 to make a standard process for processing JSON in Java files. Before it was made programmers had to use third-party libraries like Jackson and Gson. This API was made to be efficient while offering both the tree based and stream-based processing models.

Later, JSR 367 was implemented into Java EE 8 offering a more binding API for the conversion of Java objects into JSON.

**Downloading JSR 353 JAR files**

JSR 353 is in Java EE distributions, but standalone files can be downloaded from official sources. Both links are in the resource section of this paper.

**Real World Use Cases**

1. RESTful Web Services

Most modern apps use the RESTful APIs to send data between clients and servers. JSR 353 API allows the developers to swiftly parse and generate JSON when handling HTTP requests and responses. A good example of this would be a bank application using JSON to format the account details in response to web clients.

1. Configuration Management

Applications normally store their configuration settings into JSON files. Using the object model API, programmers can save, load, and adjust the configurations dynamically. For example, a cloud service system preference in a JSON file that gets updated based on user settings.

1. Data Processing and Analytics

Most organizations collect and process massive amounts of data from different sources. The streaming API allows for them to efficiently parse the large datasets without loading all of it to memory.

1. IoT (Internet of Things)

IoT devices normally communicate with lightweight datasets like JSON. This API helps developers parse incoming data and generate responses quickly.

1. Integrations with NoSQL Databases

Databases such as MongoDB store data in a format that is like JSON. The JSR 353 API helps with serialization of data making the interaction across the database smoother.

All of these are real world applications that can be applied with JSON and the JSR 353 API. There is a long list of things that JSR 353 can be used for, and these are just a few hand-picked examples that highlight the usage of this API.

Resources

Crudu,A. (April,2025). Mastering JSON Data Handling in Java RESTful APIs. MoldStud.

Retrieved from: [Master JSON Data Handling in Java RESTful APIs Best Tips | MoldStud](https://moldstud.com/articles/p-mastering-json-data-handling-in-java-restful-apis-tips-and-best-practices)

Kotamraju,J. (July,2013). Java API for JSON Processing: An Introduction to JSON. Oracle.

Retrieved from: [Java API for JSON Processing](https://www.oracle.com/java/technologies/json.html)

JSR 353 Official Page- [The Java Community Process(SM) Program - communityprocess - final](https://jcp.org/aboutJava/communityprocess/final/jsr353/index.html)

Java Community Process Page- [The Java Community Process(SM) Program - JSRs: Java Specification Requests - detail JSR# 353](https://jcp.org/en/jsr/detail?id=353)