## C:\Users\kaveti_s\Desktop\icons-facebook-facebook-icon-social-media-icon-ios-7-facebook-png--9.png

## JSON PARSER TUTORIAL

Jackson JSON

/[smlcodes](https://twitter.com/smlcodes) [/smlcodes](https://www.facebook.com/SmlCodes-216971275364320/) [/smlcodes](https://www.youtube.com/channel/UCO0JjZa8BcRvc3rTe0BL9HA)



***A*** [***SmlCodes.Com***](http://www.smlcodes.com/) ***Small presentation***

In Association with Idleposts.com

## Jackson JSON Tutorial



**Small Codes**

Programming Simplified



Copyright © 2016 SmlCodes.com All rights Reserved.

**For more tutorials & Articles visit** [**SmlCodes.com**](http://www.smlcodes.com/)

**Copyright © 2016 Smlcodes.com**

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, **without the prior written permission** of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

**Every effort has been made in the preparation of this book to ensure the accuracy of the information presented**. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, SmlCodes.com, nor its dealers or distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

**Smlcodes.com has endeavored to provide trademark information** about all the companies and products mentioned in this book by the appropriate use of capitals. However, SmlCodes.com Publishing cannot guarantee the accuracy of this information.

If you discover any errors on our website or in this tutorial, please notify us at [support@smlcodes.com](mailto:support@smlcodes.com) or [smlcodes@gmail.com](mailto:smlcodes@gmail.com)

**First published on** Dec 2016, Published by **SmlCodes.com**

Author Credits

Name : **Satya Kaveti**

Email : [satyakaveti@gmail.com](mailto:satyakaveti@gmail.com)

Website : smlcodes.com, satyajohnny.blogspot.com

Digital Partners



[  1](#_Toc471745330)

[JSON PARSER TUTORIAL 1](file:///C:\Users\kaveti_s\Desktop\Jackson%20Json%20Tutorial.docx#_Toc471745331)

[Jackson JSON Tutorial 1](#_Toc471745332)

[Jackson Json Tutorial 4](#_Toc471745333)

[1. Data Binding Way 4](#_Toc471745334)

[2. Tree Model Way 7](#_Toc471745335)

[3. Streaming API 8](#_Toc471745336)

# Jackson Json Tutorial

Jackson is a very popular and efficient Java-based library to serialize or map Java objects to JSON and vice versa. We have following other JSON Parsers in java like

* **Jackson**
* **GSON**
* **Boon**
* **JSON.org**
* **JSONP**

To use Jackson JSON Java API in our project, we can add it to the project build path or if you are using maven, we can add below dependency. Or you can [**download**](http://central.maven.org/maven2/com/fasterxml/jackson/core/jackson-core/2.2.3/jackson-core-2.2.3.jar)directly from [**here.**](http://central.maven.org/maven2/com/fasterxml/jackson/core/jackson-core/2.2.3/jackson-core-2.2.3.jar)

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-core -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

<version>2.2.3</version>

</dependency>

Jackson provides three different ways to process JSON

**1. Data Binding** **−** It reads and writes JSON content as discrete events. JsonParser reads the data, whereas JsonGenerator writes the data.

**2. Tree Model** − It prepares an in-memory tree representation of the JSON document. ObjectMapper build tree of JsonNode nodes. It is most flexible approach. It is analogous to DOM parser for XML.

**3. Streaming API** − It converts JSON to and from Plain Old Java Object (POJO) using property accessor or using annotations. ObjectMapper reads/writes JSON for both types of data bindings. Data binding is analogous to **JAXB parser** for XML

## 1. Data Binding Way

We are using **data binding** way to convert Java object to JSON & then JSON to Java Object.

* To convert Java Object to JSON file we use writeValue(...)
* Convert JSON file data to Java object, readValue(...)

### Example : convert Java object to JSON

It will do 3 things here

1. Read JSON File location
2. Convert Java Object to JSON String
3. Convert JSON String to JSON data & Save in .json file

1. Create a POJO Class that contains JSON attaributes as Data menmbers

**package** databinding;

**import** java.util.List;

**public** **class** StudentBo {

**private** **int** id;

**private** String name;

**private** List<String> address;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** List<String> getAddress() {

**return** address;

}

**public** **void** setAddress(List<String> address) {

**this**.address = address;

}

}

2.Write java programe to convert Java object to JSON

**package** databinding;

**import** java.io.File;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.codehaus.jackson.map.ObjectMapper;

**public** **class** JavaToJSON {

**public** **static** **void** main(String[] args) {

StudentBo studentBo = **new** StudentBo();

studentBo.setId(101);

studentBo.setName("Satya Kaveti");

List<String> adr = **new** ArrayList<>();

adr.add("D.No:3-100");

adr.add("NEAR RAMALAYAM");

adr.add("VIJAYAWADA");

adr.add("PINCODE:520007");

studentBo.setAddress(adr);

**try**{

ObjectMapper mapper = **new** ObjectMapper();

mapper.writeValue(**new** File("D:\\JavaToJSON.json"), studentBo);

//Convert object to JSON string

String jsonInString = mapper.writeValueAsString(studentBo);

System.***out***.println("Converted object to JSON string \n"+jsonInString);

//Convert object to JSON string and pretty print

jsonInString = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(studentBo);

System.***out***.println("JSON :\n"+jsonInString);

System.***out***.println("JAVA TO JSON COMPLETED !!");

}**catch** (Exception e) {

e.printStackTrace();

}

}

}

3. After Running the programe we get following Output. & file will store in given location

Converted object to JSON string

{"id":101,"name":"Satya Kaveti","address":["D.No:3-100","NEAR RAMALAYAM","VIJAYAWADA","PINCODE:520007"]}

JSON :

{

"id" : 101,

"name" : "Satya Kaveti",

"address" : [ "D.No:3-100", "NEAR RAMALAYAM", "VIJAYAWADA", "PINCODE:520007" ]

}

JAVA TO JSON COMPLETED !!

### Example: JSON to Java Object

It will do 3 things here

1. Read JSON File location
2. Convert JSON File Data to JSON String
3. Convert JSON String to Java Object

**package** databinding;

**import** java.io.File;

**import** org.codehaus.jackson.map.ObjectMapper;

**public** **class** JsonToJava {

**public** **static** **void** main(String[] args) {

ObjectMapper mapper = **new** ObjectMapper();

**try** {

// Convert JSON string from file to Object

StudentBo StudentBo = mapper.readValue(**new** File("C:\\Users\\kaveti\_s\\Desktop\\JSONFIles\\JavaToJSON.json"),

StudentBo.**class**);

System.***out***.println(StudentBo.getId());

System.***out***.println(StudentBo.getName());

System.***out***.println(StudentBo.getAddress());

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

Output

101

Satya Kaveti

[D.No:3-100, NEAR RAMALAYAM, VIJAYAWADA, PINCODE:520007]

## 2. Tree Model Way

 We can use **“Tree Model”** to represent JSON, and perform the read and write operations via JsonNode, it is similar to an XML DOM tree.

This **tree.json** file is used for TreeModel Traversing Example

{

"id" : 1,

"name" : {

"first" : "Satya",

"last" : "Kaveti"

},

"contact" : [

{ "type" : "phone/home", "ref" : "040-2581859"},

{ "type" : "phone/work", "ref" : "7893640870"}

]

}

 Using Jackson TreeModel (JsonNode) to parse and traversal above JSON file

**package** treemodel;

**import** java.io.File;

**import** org.codehaus.jackson.JsonNode;

**import** org.codehaus.jackson.map.ObjectMapper;

**public** **class** TreeModelExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**try** {

**long** id;

String firstName = "";

String middleName = "";

String lastName = "";

ObjectMapper mapper = **new** ObjectMapper();

JsonNode root = mapper.readTree(**new** File("C:\\ JSONFIles\\tree.json"));

// Get id

id = root.path("id").asLong();

System.***out***.println("id : " + id);

// Get Name

JsonNode nameNode = root.path("name");

**if** (nameNode.isMissingNode()) {

// if "name" node is missing

} **else** {

firstName = nameNode.path("first").asText();

// missing node, just return empty string

middleName = nameNode.path("middle").asText();

lastName = nameNode.path("last").asText();

System.***out***.println("firstName : " + firstName);

System.***out***.println("middleName : " + middleName);

System.***out***.println("lastName : " + lastName);

}

// Get Contact

JsonNode contactNode = root.path("contact");

**if** (contactNode.isArray()) {

// If this node an Arrray?

}

**for** (JsonNode node : contactNode) {

String type = node.path("type").asText();

String ref = node.path("ref").asText();

System.***out***.println("type : " + type);

System.***out***.println("ref : " + ref);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

Output

id : 1

firstName : Satya

middleName :

lastName : Kaveti

type : phone/home

ref : 040-2581859

type : phone/work

ref : 7893640870

## 3. Streaming API

[Jackson](http://jackson.codehaus.org/) supports read and write JSON via high-performance **Jackson Streaming APIs**, or incremental mode. Read this Jackson [Streaming APIs](http://wiki.fasterxml.com/JacksonStreamingApi) document for detail explanation on the benefit of using streaming API.

Jackson’s streaming processing is high-performance, fast and convenient, but it’s also difficult to use, because you need to handle each and every detail of JSON data.

In this tutorial, we show you how to use following Jackson streaming APIs to read and write JSON data.

1. **JsonGenerator – Write to JSON.**
2. **JsonParser – Parse JSON.**

### 1. JsonGenerator Example

In this example, you use “JsonGenerator” to write JSON **“field name”, “values” and “array of values”** into a file name **“skill.json“.** See code comments for self-explanatory.

**package** streamingapi;

**import** java.io.File;

**import** org.codehaus.jackson.JsonEncoding;

**import** org.codehaus.jackson.JsonFactory;

**import** org.codehaus.jackson.JsonGenerator;

**public** **class** JsonGeneratorExample {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**try** {

JsonFactory jfactory = **new** JsonFactory();

/\*\*\* write to file \*\*\*/

JsonGenerator jGenerator = jfactory.createJsonGenerator(**new** File("C:\\skill.json"), JsonEncoding.***UTF8***);

jGenerator.writeStartObject(); // {

jGenerator.writeStringField("name", "Satya"); // "name" : "Satya"

jGenerator.writeNumberField("age", 27); // "age" : 27

jGenerator.writeFieldName("skills"); // "skills" :

jGenerator.writeStartArray(); // [

jGenerator.writeString("JAVA"); // "JAVA"

jGenerator.writeString("Struts"); // "Struts"

jGenerator.writeString("Springs"); // "Springs"

jGenerator.writeEndArray(); // ]

jGenerator.writeEndObject(); // }

System.***out***.println("Json file created !!");

jGenerator.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

If we open the created JSON file, it will like below formate

{

"name": "Satya",

"age": 27,

"skills": [

"JAVA",

"Struts",

"Springs"

]

}

### 2. JsonParser Example

JsonParser is used to parse or read above file “skill.json“, and display each of the values.

In streaming mode, every JSON **“string”** is consider as a single token, and each tokens will be processed incremental, that why we call it **“incremental mode”.** For example,

{

"name":"Satya"

}

1. Token 1 = “{“
2. Token 2 = “name”
3. Token 3 = “mkyong”
4. Token 4 = “}”

**package** streamingapi;

**import** java.io.File;

**import** org.codehaus.jackson.JsonFactory;

**import** org.codehaus.jackson.JsonParser;

**import** org.codehaus.jackson.JsonToken;

**public** **class** JsonParserExample {

**public** **static** **void** main(String[] args) {

**try** {

JsonFactory jfactory = **new** JsonFactory();

/\*\*\* read from file \*\*\*/

JsonParser jParser = jfactory.createJsonParser(**new** File("C:\\skill.json"));

// loop until token equal to "}"

**while** (jParser.nextToken() != JsonToken.***END\_OBJECT***) {

String fieldname = jParser.getCurrentName();

**if** ("name".equals(fieldname)) {

// current token is "name",

// move to next, which is "name"'s value

jParser.nextToken();

System.***out***.println(jParser.getText()); // display mkyong

}

**if** ("age".equals(fieldname)) {

// current token is "age",

// move to next, which is "name"'s value

jParser.nextToken();

System.***out***.println(jParser.getIntValue()); // display 29

}

**if** ("messages".equals(fieldname)) {

jParser.nextToken(); // current token is "[", move next

// messages is array, loop until token equal to "]"

**while** (jParser.nextToken() != JsonToken.***END\_ARRAY***) {

// display msg1, msg2, msg3

System.***out***.println(jParser.getText());

}

}

}

jParser.close();

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

Output

Satya

27