**Jackson Tree Model Example**

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

*P.S Tested with Jackson 2.6.3*

1. TreeModel Traversing Example

1.1 JSON file, top level represents an object.

c:\\user.json

{

"id" : 1,

"name" : {

"first" : "Yong",

"last" : "Mook Kim"

},

"contact" : [

{ "type" : "phone/home", "ref" : "111-111-1234"},

{ "type" : "phone/work", "ref" : "222-222-2222"}

]

}

1.2 Using Jackson TreeModel (JsonNode) to parse and traversal above JSON file. Read comments for self-explanatory.

JacksonTreeModel.java

package com.mkyong.json;

import java.io.File;

import java.io.IOException;

import com.fasterxml.jackson.core.JsonGenerationException;

import com.fasterxml.jackson.databind.JsonMappingException;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

public class JacksonTreeModel {

public static void main(String[] args) {

try {

long id;

String firstName = "";

String middleName = "";

String lastName = "";

ObjectMapper mapper = new ObjectMapper();

JsonNode root = mapper.readTree(new File("c:\\user.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 (JsonGenerationException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Output

id : 1

firstName : Yong

middleName :

lastName : Mook Kim

type : phone/home

ref : 111-111-1234

type : phone/work

ref : 222-222-2222

2. TreeModel Traversing Example – Part 2

2.1 JSON file, top level represents an Array.

c:\\user2.json

[

{

"id" : 1,

"name" : {

"first" : "Yong",

"last" : "Mook Kim"

},

"contact" : [

{ "type" : "phone/home", "ref" : "111-111-1234"},

{ "type" : "phone/work", "ref" : "222-222-2222"}

]

},

{

"id" : 2,

"name" : {

"first" : "Yong",

"last" : "Zi Lap"

},

"contact" : [

{ "type" : "phone/home", "ref" : "333-333-1234"},

{ "type" : "phone/work", "ref" : "444-444-4444"}

]

}

]

2.2 The concept is same, just loops the first node.

ObjectMapper mapper = new ObjectMapper();

JsonNode rootArray = mapper.readTree(new File("c:\\user2.json"));

for(JsonNode root : rootArray){

//refer example 1.2 above, same ways to process nodes

}

3. TreeModel CRUD Example

3.1 This example, show you how to create, update and remove nodes (ObjectNode and ArrayNode). Read the comments for self-explanatory.

JacksonTreeModel.java

package com.mkyong.json;

import java.io.File;

import java.io.IOException;

import com.fasterxml.jackson.core.JsonGenerationException;

import com.fasterxml.jackson.databind.JsonMappingException;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.databind.node.ArrayNode;

import com.fasterxml.jackson.databind.node.ObjectNode;

public class JacksonTreeModel {

public static void main(String[] args) {

try {

ObjectMapper mapper = new ObjectMapper();

JsonNode root = mapper.readTree(new File("c:\\user.json"));

String resultOriginal = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(root);

System.out.println("Before Update " + resultOriginal);

// 1. Update id to 1000

((ObjectNode) root).put("id", 1000L);

// 2. If middle name is empty , update to M

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

if ("".equals(nameNode.path("middle").asText())) {

((ObjectNode) nameNode).put("middle", "M");

}

// 3. Create a new field in nameNode

((ObjectNode) nameNode).put("nickname", "mkyong");

// 4. Remove last field in nameNode

((ObjectNode) nameNode).remove("last");

// 5. Create a new ObjectNode and add to root

ObjectNode positionNode = mapper.createObjectNode();

positionNode.put("name", "Developer");

positionNode.put("years", 10);

((ObjectNode) root).set("position", positionNode);

// 6. Create a new ArrayNode and add to root

ArrayNode gamesNode = mapper.createArrayNode();

ObjectNode game1 = mapper.createObjectNode();

game1.put("name", "Fall Out 4");

game1.put("price", 49.9);

ObjectNode game2 = mapper.createObjectNode();

game2.put("name", "Dark Soul 3");

game2.put("price", 59.9);

gamesNode.add(game1);

gamesNode.add(game2);

((ObjectNode) root).set("games", gamesNode);

// 7. Append a new Node to ArrayNode

ObjectNode email = mapper.createObjectNode();

email.put("type", "email");

email.put("ref", "abc@mkyong.com");

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

((ArrayNode) contactNode).add(email);

String resultUpdate = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(root);

System.out.println("After Update " + resultUpdate);

} catch (JsonGenerationException e) {

e.printStackTrace();

} catch (JsonMappingException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

}

Output

Before Update {

"id" : 1,

"name" : {

"first" : "Yong",

"last" : "Mook Kim"

},

"contact" : [ {

"type" : "phone/home",

"ref" : "111-111-1234"

}, {

"type" : "phone/work",

"ref" : "222-222-2222"

} ]

}

After Update {

"id" : 1000,

"name" : {

"first" : "Yong",

"middle" : "M",

"nickname" : "mkyong"

},

"contact" : [ {

"type" : "phone/home",

"ref" : "111-111-1234"

}, {

"type" : "phone/work",

"ref" : "222-222-2222"

}, {

"type" : "email",

"ref" : "abc@mkyong.com"

} ],

"position" : {

"name" : "Developer",

"years" : 10

},

"games" : [ {

"name" : "Fall Out 4",

"price" : 49.9

}, {

"name" : "Dark Soul 3",

"price" : 59.9

} ]

}