**Jackson API**: Jackson API provides number of methods to work with JSON data.

Using Jackson API:

1. We can convert Java object to JSON string.
2. Reform the Object from JSON string.

Maven dependency:

<dependency>

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

         <artifactId>jackson-core</artifactId>

         <version>2.6.3</version>

</dependency>

<dependency>

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

         <artifactId>jackson-databind</artifactId>

         <version>2.6.3</version>

</dependency>

1. Create **JSON** from **Java object**

Three ways:

* + Using ObjectMapper.
  + Using JsonNode Tree.
  + Using JSON stream.

1. using ObjectMapper:

ObjectMapper class is used to convert Java object to JSON and JSON to java object.

Step 1: Instantiate ObjectMapper class

ObjectMapper mapper=new ObjectMapper();

Step 2: call any write methods to convert Java Object to Json

map.writeValue(System.out, emp.class);

[note there should be POJO class with getters and setters methods in it].

1. Using JsonNode Tree: can prepare JSON document on fly. No need of POJO class.

Step 1: Define JsonNodeFactory, ObjectMapper classes.

JsonNodeFactory factory=new JsonNodeFactory(false);

ObjectMapper mapper=new ObjectMapper();

Step 2: Get ObjectNode from JsonNodeFactory.

ObjectNode emp=factory.objectNode();

Step 3: use put, add and set methods Object node to populate data.

emp.put(“id”,12);

emp.put(“fname”,”chakra”);

emp.put(“lname”,”balla”);

For arrays: Get ArrayNode from JsonNodeFactory and use add method.

ArrayNode hobbies=factory.ArrayNode();

hobbies.add(“swimming”,”reading”,”cooking”);

emp.set(“hobbies”,hobbies);

Step 4: use mapper.writeValueAsString(emp)

sout(mapper.writeValueAsString(emp));

1. Using JsonGenerator. No POJO class required.

Step 1: Initialize JsonFactory

JsonFactory factory=new JsonFactory();

Step 2: Get JsonGenerator form JsonFactory instance.

JsonGenerator generator=factory.createGenerator(new FileWriter(new File(“emp.json”)));

Step 3: User write methods of generator objects for writing json data.

generator.writeStartObject();

generator.writeFieldName(“id”);

generator.writeString(“12”);

For arrays:

generator.writeFieldName(“hobbies”);

generator.writeStartArray();

generator.writeString(“swimming”);

generator.writeString(“cooking”);

End the array and end object

generator.writeEndArray();

generator.writeEndObject();

Step 4: Finally, close generator.

generator.close()

1. Parsing JSON data:

Step 1: Instantiate JsonFactory class.

JsonFactory factory =new JsonFactory();

Step 2: Get the JsonParser instance from factory.

JsonParser parser=factory.createParser(new File(“emp.json”));

Step 3: Use parse API’s line nextToken, getToken and getTest() to process the json file.

1. Pretty print JSON: By enabling ‘SerializationFeatue.INDENT\_OUTPUT’.

ObjectMapper mapper=new ObjectMapper();

Mapper.enable(SerializationFeatue.INDENT\_OUTPUT);

String json=mapper.writeValueAsString(emp);

1. Ignore Empty, null fields

* By configuring ObjectMapper

ObjectMapper mapper=new ObjectMapper();

mapper.setSerializationInclusion(Include.NON\_EMPTY);

* By using JsonInclude annotation

@JsonInclude(Include.NON\_EMPTY) -- above POJO class.

1. Serialize the order of properties. Using @JsonPropertyOrder annotation above POJO class.

@JsonPropertyOrder({“fname”,”lname”,”hoobies”,”id”})

1. Change the field names in JSON serialization. Using @JsonProperty to the variables in POJO class.

@JsonProperty(“EMPLOYEE\_ID”)

private int id;

1. Ignoring fields while doing json serialization:

@JsonIgnoe

Private int id;

1. JsonAnyGetter: contents of the map are serialized like actual properties of the object.

@JsonAnyGetter

1. JsonAnySetter:if any unreconized field is identified, then this method is called.

@JsonAnySetter.

1. Convert JSON to object: Requires POJO class.

ObjectMapper mapper=new ObjectMapper();

String json=mapper.writeValueAsString(emp);

Employee emp=mapper.readValue(json,Employee.class);

1. Convert JSON to Map: use TypeReference . require POJO class.

ObjectMapper mapper=new ObjectMapper();

String json=mapper.writeValueAsString(emp);

Map<String, Employee> map=mapper.reafValue(json, new TypeReference<Map<String, Employee>>(){});

**import** **java.util.ArrayList**;

**import** **java.util.List**;

**public** **class** **Employee** {

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

**private** String firstName;

**private** String lastName;

**private** List<String> hobbies = **new** ArrayList<>();

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

**return** id;

}

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

**this**.id = id;

}

**public** String **getFirstName**() {

**return** firstName;

}

**public** **void** **setFirstName**(String firstName) {

**this**.firstName = firstName;

}

**public** String **getLastName**() {

**return** lastName;

}

**public** **void** **setLastName**(String lastName) {

**this**.lastName = lastName;

}

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

**return** hobbies;

}

**public** **void** **setHobbies**(List<String> hobbies) {

**this**.hobbies = hobbies;

}

**@Override**

**public** String **toString**() {

StringBuilder builder = **new** StringBuilder();

builder.append("Employee [id=").append(id).append(", firstName=")

.append(firstName).append(", lastName=").append(lastName)

.append(", hobbies=").append(hobbies).append("]");

**return** builder.toString();

}

}

**public** **class** **Main** {

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

Employee emp = **new** Employee();

emp.setFirstName("Hari Krishna");

emp.setId(**1**);

emp.setLastName("Gurram");

emp.getHobbies().add("Trekking");

emp.getHobbies().add("Blogging");

emp.getHobbies().add("Cooking");

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

mapper.writeValue(System.out, emp);

}

}

**public** **class** **Main** {

**public** **static** **void** **main**(String args[]) **throws** JsonGenerationException,

JsonMappingException, IOException {

Employee emp = **new** Employee();

emp.setFirstName("Hari Krishna");

emp.setId(**1**);

emp.setLastName("Gurram");

emp.getHobbies().add("Trekking");

emp.getHobbies().add("Blogging");

emp.getHobbies().add("Cooking");

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

String json = mapper.writeValueAsString(emp);

System.out.println(json);

System.out.println("Deserializing JSON to Object");

Employee emp1 = mapper.readValue(json, Employee.class);

System.out.println(emp1.getId() + " " + emp1.getFirstName() + " "

+ emp1.getLastName() + " " + emp1.getHobbies());

}

}