## Reference links

<https://www.baeldung.com/jackson>

<https://github.com/eugenp/tutorials/tree/master/jackson-simple>

All annotations descriptions are here

<https://www.tutorialspoint.com/jackson_annotations/jackson_annotations_jsonpropertyorder.htm>

https://www.baeldung.com/jackson-object-mapper-tutorial

## Dependencies

<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-core</artifactId>  
 <version>2.16.0</version>  
</dependency>  
<dependency>  
 <groupId>com.fasterxml.jackson.core</groupId>  
 <artifactId>jackson-annotations</artifactId>  
 <version>2.16.0</version>  
</dependency>

<dependency>

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

<artifactId>jackson-databind</artifactId>

<version>2.16.0</version>

</dependency>

# Json to java pojo

1. U should write a json schema file like below using this only we can generate the java pojos

{

"type": "object",

"javaType": "guru.learningjournal.kafka.examples.types.LineItem",

"properties": {

"ItemCode": {"type": "string"},

"ItemDescription": {"type": "string"},

"ItemPrice": {"type": "number"},

"ItemQty": {"type": "integer"},

"TotalValue": {"type": "number"}

}

}

1. U should add this plugin to generate the java classes from schema files using below maven code

<plugin>

<groupId>org.jsonschema2pojo</groupId>

<artifactId>jsonschema2pojo-maven-plugin</artifactId>

<version>0.5.1</version>

<executions>

<execution>

<goals>

<goal>generate</goal>

</goals>

<configuration>

<sourceDirectory>${project.basedir}/src/main/resources/schema/</sourceDirectory>

<outputDirectory>${project.basedir}/src/main/java/</outputDirectory>

<includeAdditionalProperties>false</includeAdditionalProperties>

<includeHashcodeAndEquals>false</includeHashcodeAndEquals>

<generateBuilders>true</generateBuilders>

</configuration>

</execution>

</executions>

</plugin>

# **Json annotations**

1. @JsonInclude(Include.NON\_NULL)

class Foo

{

String bar;

}

[NON\_NULL](https://fasterxml.github.io/jackson-annotations/javadoc/2.6/com/fasterxml/jackson/annotation/JsonInclude.Include.html#NON_NULL)

Value that indicates that only properties with non-null values are to be included means null values will be ignored

Or

public class Foo {

@JsonInclude(JsonInclude.Include.NON\_NULL)

private String fieldOne;

@JsonInclude(JsonInclude.Include.ALWAYS)

private String fieldTwo;

}

ObjectMapper mapper = new ObjectMapper();

mapper.configure(SerializationFeature.WRITE\_NULL\_MAP\_VALUES, false);

**ObjectMapper** mapper = **new** **ObjectMapper**(); mapper.setSerializationInclusion(Include.NON\_NULL);

Example

@JsonInclude(JsonInclude.Include.*NON\_NULL*)  
class Employee {  
 @JsonProperty("employeeName") //Due to this annotation while converting from object to json the fields will be renamed to employeeName instead of variable name  
 String empname;  
 Integer empid;  
 String address;

}

ObjectMapper mapper = new ObjectMapper();  
Employee emp=new Employee("manideep",12,null);//Here we are passing null for address, as we ignored null fields this null value won’t appear in console  
String json = mapper.writeValueAsString(emp);  
System.out.println(json);

Output

{"empid":12,"employeeName":"manideep"}

1. @JsonPropertyOrder

@JsonPropertyOrder({ "rollNo", "name" })

class Student {

private String name;

private int rollNo;

}

1. @JsonProperty(“customName other than variable name”)

## Convert object to string

**While converting make sure the target class should have getters and setters**

**ObjectMapper** mapper = **new** **ObjectMapper**(); **MyDto** dtoObject = **new** **MyDto**(); **String** dtoAsString = mapper.writeValueAsString(dtoObject);

## Convert String to object

ObjectMapper mapper = new ObjectMapper();

String json = "{\"id\" : 1}";

Student student = mapper.readerFor(Student.class).readValue(json);

System.out.println(student.getTheId());

## Convert obj to byte array

byte[] b= objectMapper.writeValueAsBytes(empObject);

if this is not there then 1st we will convert from object to string then string to bytes

= mapper.writeValueAsString(dtoObject).getBytes();