Jackson

Json-b api

If you need to match JSON object fields to Java object fields in a different way, you need to either use a custom serializer and deserializer, or use some of the many [**Jackson Annotations**](http://tutorials.jenkov.com/java-json/jackson-annotations.html).

## **Without Annotations:**

Without any annotations, it does what is called POJO mapping, it just uses [reflection](http://docs.oracle.com/javase/tutorial/reflect/member/index.html) on the instance members and uses some rules about how to map the keys in the json to the names of the instance members. \*note: it works on *private* members as well as *public* or *package protected* as well

If it doesn't match the names of the instance members, then it starts trying to match the getXXX and setXXX methods, if it doesn't match anything then it gives up.

## **With Annotations:**

It uses the metadata supplied by the annotations to do the mapping and conversions.

It is always better to explicitly use the annotations when you have the source to add them to, then there is no guess work on what gets mapped to what.

Remember explicit is always better than implicit!

@JsonView is used to **indicate the View** in which the property will be included for serialization/deserialization.

## Custom Serializer

Sometimes you want to serialize a Java object to JSON differently than what Jackson does by default. For instance, you might want to use different field names in the JSON than in the Java object, or you might want to leave out certain fields altogether.

Jackson enables you to set a custom serializer on the ObjectMapper. This serializer is registered for a certain class, and will then be called whenever the ObjectMapper is asked to serialize a Car object. Here is an example that shows how to register a custom serializer for the Car class:

To read Java objects from JSON with Jackson properly, it is important to know how Jackson maps the fields of a JSON object to the fields of a Java object, so I will explain how Jackson does that.

By default Jackson maps the fields of a JSON object to fields in a Java object by matching the names of the JSON field to the getter and setter methods in the Java object. Jackson removes the "get" and "set" part of the names of the getter and setter methods, and converts the first character of the remaining name to lowercase.

Json-b

**JSON-B** expects all classes to have a no-argument public constructor, which it uses to construct class instances.