Search...

JSON using Jackson in REST API Implementation with Spring Boot

Last Updated : 23 Jul, 2025

When we build REST APIs with **Spring Boot,** we need to exclude **NULL** values from the JSON responses. This is useful when we want to optimize the data being transferred, making the response more compact and easier to process for the client.

In this article, we are going to learn the approach that is used to implement the externalization of ON/OFF feature using **Jackson – A Java-based library to serialize or map Java objects to JSON and vice versa.**

Unsuccessful Approaches

Initially, we tried using Spring Boot's readily available approaches, which were not successful, such as:

1. Using spring.jackson.default-property-inclusion in application.properties

We tried setting the **spring.jackson.default-property-inclusion** property in the **application.properties** file. This property accepts values such as always, **non_null, non_absent, non_default**, and **non_empty**. This approach did not work as expected.

> *spring.jackson.default-property-inclusion=non_null*

Note: This approach should work if configured correctly. We just need to ensure that the property is placed in the correct application.properties file and that the application is properly restarted after making changes.

2. Extending WebMvcConfigurationSupport to Customize the ObjectMapper

We attempted to extend the **WebMvcConfigurationSupport** class and customize the **ObjectMapper** as shown below:

```
@Configuration
class WebMvcConfiguration extends WebMvcConfigurationSupport {
    @Override
    protected void extendMessageConverters(List<HttpMessageConverter<?>>
converters) {
        for (HttpMessageConverter<?> converter : converters) {
            if (converter instanceof MappingJackson2HttpMessageConverter) {
                ObjectMapper mapper = ((MappingJackson2HttpMessageConverter)
converter).getObjectMapper();
                mapper.setSerializationInclusion(Include.NON_NULL);
            }
        }
    }
}
```

Note: It is generally recommended to use **WebMvcConfigurer** instead of **WebMvcConfigurationSupport** to avoid disabling some of Spring Boot's auto-configurations.

Using Jackson2ObjectMapperBuilderCustomizer

Now, Creating an instance of ***org.springframework.http.converter.json.Jackson2ObjectMapperBuilderCustomizer.*** Below the given code is for your reference.

```
@Bean
Public Jackson2ObjectMapperBuilderCustomizer customJackson(){
    return new Jackson2ObjectMapperBuilderCustomizer(){
    @Override
    public void customize(Jackson2ObjectMapperBuilder builder){
        builder.serializationInclusion(Include.NON_NULL);
        builder.failonUnknownProperties(false);
        }
    };
}
```

Note: This approach is correct and should work as expected. But it does not provide the flexibility to externalize the feature (i.e., turn it ON/OFF dynamically).

Successful Approach

To achieve the desired functionality, we implemented a solution that allows the feature to exclude **NULL** values in the JSON response. Below are the steps:

Step 1: Create a JSON Response Object

Create a JSON Response Object, if not created. This is the object which while serializing to JSON you want to '***Exclude/Include Null fields***' feature based on the property in ***application.properties***. You can see below the Response object for your reference.

```
public class RegistrationResponse{

    @JsonProperty("success")
    private Boolean success = null;

    @JsonProperty("message")
    private String message = null;

    @JsonProperty("data")
    private String data = null;

    @JsonProperty("error_code")
    private Integer errorCode = null;

    public RegistrationResponse success(Boolean success){
        this.success = success;
        return this;
        }

        @NotNull
        public Boolean isSuccess(){
            return success;
            }
        public void setSuccess(Boolean success){
            this.success = success;
```

```
            this.message = message;
            return this;
            }

        @NotNull
        public String getMessage(){
            return message;
        }

        public void setMessage(String message){
```

```
        this.message = message;
    }
```

Step 2: Externalize the Configuration

Create a property by the name '***config.response.json.format.exclude_null***' which can take values as '***true***' or '***false***'. The value '***true***' means to exclude null fields in the JSON Response & the value '***false***' means to not exclude null fields. Also, bind this property to a class-level property so that the value can be read, Below given code is for your reference.

> *config.response.json.format.exclude_null = true*
>
> *@Value("${config.response.json.format.exclude_null}")*
> *private boolean toExcludeNull;*

Step 3: Implement the Logic to Exclude NULLs

Implement the actual logic to exclude nulls based on the global property value by - Creating an instance of ***com.fasterxml.jackson.databind.ObjectMapper***.

- Setting ***SerializationInclusion*** property of the created mapper to ***Include.NON_NULL*** if the global property value is true, otherwise set to ***Include.ALWAYS***.
- Serialize the Response object to a String using the mapper & return the String. ***Make sure to handle JsonProcessingException***.

Let's see below for the actual code snippet to be added while constructing the JSON Response as part of the API implementation.

```
RegistrationResponse regResp = new RegistrationResponse();

regResp.setSuccess(true);
regResp.setErrorCode(null);
regResp.setData("testdata");
regResp.setMessage("success");
```

```
ObjectMapper mapper = new ObjectMapper()
mapper.enable(SerializationFeature.INDENT_OUTPUT);

if(toExcludeNull) mapper.setSerializationInclusion(Include.NON_NULL);
else mapper.serializationInclusion(Include.ALWAYS);

String regRespStr = null;
try{

    regRespStr = mapper.writeValueAsString(regResp);
    }
    catch(JsonProcessingException e)
        e.printStackTrace();
        }
    System.out.println("Formatted JSON Response:" + regRespStr);
```

Step 4: Verify the Output

The **regRespStr** variable will contain the JSON string with NULL values excluded or included based on the property value. For example:

- **If config.response.json.format.exclude_null=true,** the output will exclude NULL fields.
- **If config.response.json.format.exclude_null=false,** the output will include NULL fields.

Comment More info Advertise with us