1. Goals

The goal of this lesson is to switch from Jackson as the default solution for JSON in Spring, to using the new JSON Binding API.

2. Lesson Notes

2.1. Overview, History

JSR 374 introduced JSPN-P for JSON parsing and processing - but that is only focused on parsing, and doesn't support binding to/from POJOs.

Binding functionality is introduced with JSR 367 - JSON-B - which is, simply put, focused on converting Java objects to/from JSON messages.

The reference implementation of JSR 367 is Yasson from Eclipse; Apache Johnzon is also a solid alternative.

Keep in mind that, while Spring 5 now supports JSON-B, Jackson is still the default.

Alright, let's have a look at the functionality.

2.2. With Spring Boot

The starting point of the lesson is: m13-l3-start

We'll drive everything with a simple MVC test: *SpringHttpMessageConverterMockTest*.

First, let's check out the default JSON Http message converter, by adding a breakpoint in *AbstractMessageConverterMethodProcessor.writeWithMessageConverters()*

We can now debug the test and verify that Spring will use the*MappingJackson2HttpMessageConverter* - as we're expecting.

Next, let's actually configure JSON-B message converter - in *um-webapp:*

HTML

```
<dependency>
    <groupId>org.eclipse</groupId>
    <artifactId>yasson</artifactId>
    <version>1.0.1</version>
</dependency>

<dependency>
    <groupId>org.glassfish</groupId>
    <artifactId>javax.json</artifactId>
    <version>1.1.2</version>
</dependency>
```

And let's now add - in the *application.properties* of the project:

spring.http.converters.preferred-json-mapper=jsonb

Finally, in *Privilege.java* - let's replace *@JsonIgnore* with *@JsonbTransient* - so that we aren't using any Jackson-specific annotation.

Now, we're ready to debug the test again and make sure we're actually using the JSON-B converter.

And we're done - we have successfully switched the default Jackson HTTP Message converter to the one backed by JSON-B.

2.3. Without Spring Boot

The starting point here is: m13-l3-noboot-start

Our goal is the same here.

The setup steps are also the same as in the earlier Boot section.

The one core difference is - we'll have to create and wire in the new message converter manually:

- in *UmWebConfig.java*:

JAVA

```
public void extendMessageConverters(List<HttpMessageConverter<?>> converters) {
    JsonbHttpMessageConverter jsonbHttpMessageConverter = new JsonbHttpMessageConverter();

    converters.add(0, jsonbHttpMessageConverter);
}
```

Simply put, this creates the JSON-B converter and adds it to the top of list.

Now, if we debug the same test again, we'll see that Spring does use the new JSON-B converter, as we are expecting.

And we're done - this is the only real difference of setting up the new converter without the help of Spring Boot.

3. Resources

- Java API for JSON Binding