链滴

Spring Cloud Feign 配置 FastJson

作者：kangaroo1122
原文链接：https://ld246.com/article/1585455011524
来源网站：链滴

项目集成FastJson解析框架，一般都会添加一个配置文件，如下：

```
@Configuration
public class CustomFastJsonConfig {

    @Bean
    FastJsonHttpMessageConverter fastJsonHttpMessageConverter(){
        //1.需要定义一个convert转换消息的对象
        FastJsonHttpMessageConverter converter = new FastJsonHttpMessageConverter();
        //2:添加fastJson的配置信息
        FastJsonConfig fastJsonConfig = new FastJsonConfig();
        fastJsonConfig.setSerializerFeatures(
                // 是否输出值为null的字段,默认为false
                SerializerFeature.WriteMapNullValue,
                // 将Collection类型字段的字段空值输出为[]
                SerializerFeature.WriteNullListAsEmpty,
                // 将字符串类型字段的空值输出为空字符串
                SerializerFeature.WriteNullStringAsEmpty,
                // 将数值类型字段的空值输出为0
                SerializerFeature.WriteNullNumberAsZero,
                //Boolean字段如果为null,输出为false,而非null
                SerializerFeature.WriteNullBooleanAsFalse,
                //SerializerFeature.WriteDateUseDateFormat,
                //枚举字段输出为枚举值
                SerializerFeature.WriteEnumUsingToString,
                // 禁用循环引用
                SerializerFeature.DisableCircularReferenceDetect);
        //这个日期格式是全局格式：yyyy-MM-dd HH:mm:ss
        fastJsonConfig.setDateFormat("yyyy-MM-dd HH:mm:ss");
        fastJsonConfig.setCharset(StandardCharsets.UTF_8);

        //3处理中文乱码问题
```

```
        List<MediaType> fastMediaTypes = new ArrayList<>();
        fastMediaTypes.add(MediaType.APPLICATION_JSON);

        //4.在convert中添加配置信息.
        converter.setSupportedMediaTypes(fastMediaTypes);
        converter.setFastJsonConfig(fastJsonConfig);
        return converter;
    }
}
```

但是，当Spring Cloud微服务项目使用Feign的时候，有一个比较坑的地方，就是，Feign并不共用Spring MVC的消息转换器链，而且它默认使用的是Jackson Json解析库，而项目使用的又是FastJson，会导致混乱，甚至出现序列化/反序列化错误。

就比如上边这个配置，A应用通过Feign调用B应用的搜索接口，接口参数是一个实体对象，其中实体象有一个Date类型的时间属性参数 downDate，那么，由于上边配置的日期格式是：yyyy-MM-dd H:mm:ss，在通过Feign调用的时候，就会产生一个异常：

```
{"timestamp":"2020-03-29T03:54:55.014+0000","status":400,"error":"Bad Request","message":
JSON parse error:
Cannot deserialize value of type `java.util.Date` from String \"2020-03-03 00:00:00\":
not a valid representation (error: Failed to parse Date value '2020-03-03 00:00:00':
Cannot parse date \"2020-03-03 00:00:00\":
while it seems to fit format 'yyyy-MM-dd'T'HH:mm:ss.SSSZ', parsing fails (leniency? null));
nested exception is com.fasterxml.jackson.databind.exc.InvalidFormatException:
Cannot deserialize value of type `java.util.Date` from String \"2020-03-03 00:00:00\":
not a valid representation (error: Failed to parse Date value '2020-03-03 00:00:00':
Cannot parse date \"2020-03-03 00:00:00\":
while it seems to fit format 'yyyy-MM-dd'T'HH:mm:ss.SSSZ', parsing fails (leniency? null)) at [
ource: (PushbackInputStream);
line: 1, column: 154] (through reference chain: com.kangaroohy.entity.bo.SearchBO[\"downDa
e\"])","path":"/api/search"}
```

因此，需要为Feign单独配置FastJson，将原来的转换器替换成FastJson的，具体配置如下：

```
import com.alibaba.fastjson.serializer.SerializerFeature;
import com.alibaba.fastjson.support.config.FastJsonConfig;
import com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter;
import feign.codec.Encoder;
import org.springframework.beans.factory.ObjectFactory;
import org.springframework.boot.autoconfigure.http.HttpMessageConverters;
import org.springframework.cloud.openfeign.support.SpringEncoder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.MediaType;
import org.springframework.http.converter.HttpMessageConverter;

import java.util.ArrayList;
import java.util.List;

/**
 * @desc: feign配置
 * @author: kangaroohy
 * @create: 2020/03/26
 */
```

```
@Configuration
public class FeignClientConfig {

    @Bean
    public Encoder feignEncoder(){
        return new SpringEncoder(feignHttpMessageConverter());
    }

    private ObjectFactory<HttpMessageConverters> feignHttpMessageConverter() {
        final HttpMessageConverters httpMessageConverters = new HttpMessageConverters(cr
ateFastJsonConverter());
        return () -> httpMessageConverters;
    }

    private HttpMessageConverter createFastJsonConverter(){
        FastJsonHttpMessageConverter fastJsonConverter = new FastJsonHttpMessageConverte
();
        List<MediaType> supportedMediaTypes = new ArrayList<>();
        supportedMediaTypes.add(MediaType.APPLICATION_JSON);
        supportedMediaTypes.add(MediaType.APPLICATION_JSON_UTF8);
        supportedMediaTypes.add(MediaType.APPLICATION_ATOM_XML);
        supportedMediaTypes.add(MediaType.APPLICATION_FORM_URLENCODED);
        supportedMediaTypes.add(MediaType.APPLICATION_OCTET_STREAM);
        supportedMediaTypes.add(MediaType.APPLICATION_PDF);
        supportedMediaTypes.add(MediaType.APPLICATION_RSS_XML);
        supportedMediaTypes.add(MediaType.APPLICATION_XHTML_XML);
        supportedMediaTypes.add(MediaType.APPLICATION_XML);
        supportedMediaTypes.add(MediaType.IMAGE_GIF);
        supportedMediaTypes.add(MediaType.IMAGE_JPEG);
        supportedMediaTypes.add(MediaType.IMAGE_PNG);
        supportedMediaTypes.add(MediaType.TEXT_EVENT_STREAM);
        supportedMediaTypes.add(MediaType.TEXT_HTML);
        supportedMediaTypes.add(MediaType.TEXT_MARKDOWN);
        supportedMediaTypes.add(MediaType.TEXT_PLAIN);
        supportedMediaTypes.add(MediaType.TEXT_XML);
        fastJsonConverter.setSupportedMediaTypes(supportedMediaTypes);

        FastJsonConfig fastJsonConfig = new FastJsonConfig();
        fastJsonConfig.setSerializerFeatures(
                // 是否输出值为null的字段,默认为false
                SerializerFeature.WriteMapNullValue,
                // 将Collection类型字段的字段空值输出为[]
                SerializerFeature.WriteNullListAsEmpty,
                // 将字符串类型字段的空值输出为空字符串
                SerializerFeature.WriteNullStringAsEmpty,
                // 将数值类型字段的空值输出为0
                SerializerFeature.WriteNullNumberAsZero,
                //Boolean字段如果为null,输出为false,而非null
                SerializerFeature.WriteNullBooleanAsFalse,
                //SerializerFeature.WriteDateUseDateFormat,
                //枚举字段输出为枚举值
                SerializerFeature.WriteEnumUsingToString,
                // 禁用循环引用
                SerializerFeature.DisableCircularReferenceDetect);
```

```
        fastJsonConverter.setFastJsonConfig(fastJsonConfig);
        return fastJsonConverter;
    }
}
```

此时，就不会发生上述的报错了。

特此记录。