

SpringBoot2.0 使用 FastJson 转换器(FastJson 1.2.49)

SpringBoot 2.0.4.RELEASE

官方文档说明

目前版本依赖 MVC 的版本为 spring-webmvc.5.0.8.RELEASE, Spring 官方文档中是这样描述的:

1.11.7. Message Converters

[Same in Spring WebFlux](#)

Customization of `HttpMessageConverter` can be achieved in Java config by overriding `configureMessageConverters()` if you want to replace the default converters created by Spring MVC, or by overriding `extendMessageConverters()` if you just want to customize them or add additional converters to the default ones.

Below is an example that adds Jackson JSON and XML converters with a customized `ObjectMapper` instead of default ones:

```
@Configuration
@EnableWebMvc
public class WebConfiguration implements WebMvcConfigurer {

    @Override
    public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {
        Jackson2ObjectMapperBuilder builder = new Jackson2ObjectMapperBuilder()
            .indentOutput(true)
            .dateFormat(new SimpleDateFormat("yyyy-MM-dd"))
            .modulesToInstall(new ParameterNamesModule());
        converters.add(new MappingJackson2HttpMessageConverter(builder.build()));
        converters.add(new MappingJackson2XmlHttpMessageConverter(builder.createXmlMapper(true).build()));
    }
}
```

https://blog.csdn.net/litte_frog

翻译: 如果您想要替换 Spring MVC 创造的默认转换器, 重写 `extendmessageconverter`, 如果您只想定制它们, 或者向默认的转换器添加额外的转换器, 那么可以在 Java 配置中实现 `HttpMessageConverter` 的定制。

自定义 MVC 配置类

根据文档中的示例, 写自己的配置类:

```
@Configuration
@EnableWebMvc
public class WebConfig implements WebMvcConfigurer {

    @Override
    public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {
        // 消息转换对象
        FastJsonHttpMessageConverter fastJsonConverter = new FastJsonHttpMessageConverter();
        // 配置信息(编码字符集和日期格式)
        FastJsonConfig config = new FastJsonConfig();
        config.setCharset(Charset.forName("UTF-8"));
        config.setDateFormat("yyyyMMdd HH:mm:ss");
        fastJsonConverter.setFastJsonConfig(config);
        List<MediaType> list = new ArrayList<>();
        list.add(MediaType.APPLICATION_JSON_UTF8);
        fastJsonConverter.setSupportedMediaTypes(list);
        // 添加到converters
        converters.add(fastJsonConverter);
    }
}
```

https://blog.csdn.net/litte_frog

测试结果成功, FastJson 生效。

Content type 'text/plain;charset=UTF-8' not supported

项目中增加自定义 MVC 配置类后, 由于指定了 `MediaType` 为 `APPLICATION_JSON_UTF8` (`application/json;charset=UTF-8`) 导致一些请求类型不为 json 格式的接口报错, 但是在没有增加自定义 `WebConfig` 类之前是不会报错的。

通过 DEBUG 跟踪至如下类:

org.springframework.web.servlet.mvc.method.annotation.AbstractMessageConverterMethodArgumentResolver
readWithMessageConverters 方法中

```
EmptyBodyCheckingHttpInputMessage message;
try {
    message = new EmptyBodyCheckingHttpInputMessage(inputMessage);

    for (HttpMessageConverter<?> converter : this.messageConverters) {
        Class<HttpMessageConverter<?>> converterType = (Class<HttpMessageConverter<?>>) converter.getClass();
        GenericHttpMessageConverter<?> genericConverter =
            (converter instanceof GenericHttpMessageConverter ? (GenericHttpMessageConverter<?>) converter : null);
        if (genericConverter != null ? genericConverter.canRead(targetType, contextClass, contentType) :
            (targetClass != null && converter.canRead(targetClass, contentType))) {
            if (logger.isDebugEnabled()) {
                logger.debug("@Read [" + targetType + "] as \"" + contentType + "\" with [" + converter + "]");
            }
            if (message.hasBody()) {
                HttpInputMessage msgToUse =
                    getAdvice().beforeBodyRead(message, parameter, targetType, converterType);
                body = (genericConverter != null ? genericConverter.read(targetType, contextClass, msgToUse) :
                    ((HttpMessageConverter<?>) converter).read(targetClass, msgToUse));
                body = getAdvice().afterBodyRead(body, msgToUse, parameter, targetType, converterType);
            }
            else {
                body = getAdvice().handleEmptyBody(body, null, message, parameter, targetType, converterType);
            }
            break;
        }
    }
}
```

https://blog.csdn.net/litte_frog

表示从转换器中读取给定的类型, 如果请求的类型与转换器中类型匹配不到, 就会跳出上图代码块, 然后抛出异常。

```
if (body == NO_VALUE) {
    if (httpMethod == null || !SUPPORTED_METHODS.contains(httpMethod) ||
        (noContentType && !message.hasBody())) {
        return null;
    }
    throw new HttpMediaTypeNotSupportedException(contentType, this.allSupportedMediaTypes);
}
```

https://blog.csdn.net/litte_frog

▼ this.messageConverters = {ArrayList@6875} size = 1
0 = {FastJsonHttpMessageConverter@6973}
https://blog.csdn.net/litte_frog

转化器集合中只有 FastJson, 所以类型不匹配是正常的。

帖两段注释:

```
/**
 * Configure the {@link HttpMessageConverter}s to use for reading or writing
 * to the body of the request or response. If no converters are added, a
 * default list of converters is registered.
 * <p><strong>Note</strong> that adding converters to the list, turns off
 * default converter registration. To simply add a converter without impacting
 * default registration, consider using the method
 * {@link #extendMessageConverters(java.util.List)} instead.
 * @param converters initially an empty list of converters
 */
default void configureMessageConverters(List<HttpMessageConverter<?>> converters) {
}

/**
 * A hook for extending or modifying the list of converters after it has been
 * configured. This may be useful for example to allow default converters to
 * be registered and then insert a custom converter through this method.
 * @param converters the list of configured converters to extend.
 * @since 4.1.3
 */
default void extendMessageConverters(List<HttpMessageConverter<?>> converters) {
}
```

https://blog.csdn.net/litte_frog

翻译: 配置@link HttpMessageConverter, 用于读或写请求或响应的主体。如果没有添加转换

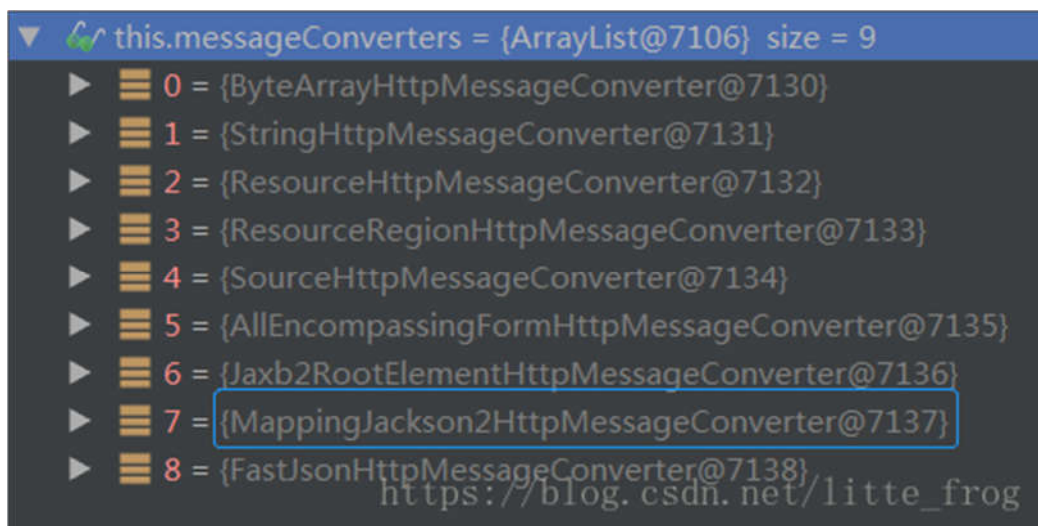
器,使用注册的转换器的默认列表。注意,在列表中添加了转换器,关闭了默认转换器注册。如果要简单地添加一个转换器而不影响默认注册,考虑使用 `extendMessageConverters` 重新编写 `WebConfig`

将 `configureMessageConverters` 方法替换为 `extendMessageConverters`

```
@Override
public void extendMessageConverters (List<HttpMessageConverter<?>> converters) {
    // 消息转换对象
    FastJsonHttpMessageConverter fastJsonConverter = new FastJsonHttpMessageConverter();
    // 配置信息 (编码字符集和日期格式)
    FastJsonConfig config = new FastJsonConfig();
    config.setCharset(Charset.forName("UTF-8"));
    config.setDateFormat("yyyy-MM-dd HH:mm:ssS");
    fastJsonConverter.setFastJsonConfig(config);
    List<MediaType> list = new ArrayList<>();
    list.add(MediaType.APPLICATION_JSON_UTF8);
    fastJsonConverter.setSupportedMediaTypes(list);
    // 添加到converters
    converters.add(fastJsonConverter);
}
```

https://blog.csdn.net/litte_frog

发送请求查看 `this.messageConverters`, 由于有 `StringHttpMessageConverter` 这个转换器, 请求类型为 `text` 可以通过, 但是 `json` 类型时 `fastjson` 并没有生效。



▼ this.messageConverters = {ArrayList@7106} size = 9

- ▶ 0 = {ByteArrayHttpMessageConverter@7130}
- ▶ 1 = {StringHttpMessageConverter@7131}
- ▶ 2 = {ResourceHttpMessageConverter@7132}
- ▶ 3 = {ResourceRegionHttpMessageConverter@7133}
- ▶ 4 = {SourceHttpMessageConverter@7134}
- ▶ 5 = {AllEncompassingFormHttpMessageConverter@7135}
- ▶ 6 = {Jaxb2RootElementHttpMessageConverter@7136}
- ▶ 7 = {MappingJackson2HttpMessageConverter@7137}
- ▶ 8 = {FastJsonHttpMessageConverter@7138}

https://blog.csdn.net/litte_frog

方式一:

Debug 查看 `FastJson` 转换器添加到集合中, 但是却没有使用, 使用的是 `MappingJackson` 这个, 所以可以对 `WebConfig` 中代码做一个小小修改, 将集合中的 `MappingJackson` 做替换。

```
@Override
public void extendMessageConverters (List<HttpMessageConverter<?>> converters) {
    // 消息转换对象
    FastJsonHttpMessageConverter fastJsonConverter = new FastJsonHttpMessageConverter();
    // 配置信息 (编码字符集和日期格式)
    FastJsonConfig config = new FastJsonConfig();
    config.setCharset(Charset.forName("UTF-8"));
    config.setDateFormat("yyyy-MM-dd HH:mm:ssS");
    fastJsonConverter.setFastJsonConfig(config);
    List<MediaType> list = new ArrayList<MediaType>();
    list.add(MediaType.APPLICATION_JSON_UTF8);
    fastJsonConverter.setSupportedMediaTypes(list);
    for (int i = 0; i < converters.size(); i++) {
        if (converters.get(i) instanceof MappingJackson2HttpMessageConverter) {
            converters.set(i, fastJsonConverter);
        }
    }
    // 添加到converters
    converters.add(fastJsonConverter);
}
```

https://blog.csdn.net/litte_frog

方式二:

如果请求类型只有 json 或者 text, 也可以还是重写 `configureMessageConverters` 方法, 添加 `StringHttpMessageConverter`

```
@Override
public void configureMessageConverters(List<HttpMessageConverter<?>> converters) {
    // 消息转换对象
    FastJsonHttpMessageConverter fastJsonConverter = new FastJsonHttpMessageConverter();
    // 配置信息(编码字符集和日期格式)
    FastJsonConfig config = new FastJsonConfig();
    config.setCharset(Charset.forName("UTF-8"));
    config.setDateFormat("yyyyMMdd HH:mm:ss");
    fastJsonConverter.setFastJsonConfig(config);
    List<MediaType> list = new ArrayList<>();
    list.add(MediaType.APPLICATION_JSON_UTF8);
    fastJsonConverter.setSupportedMediaTypes(list);

    // 支持text请求类型转string
    StringHttpMessageConverter stringConverter = new StringHttpMessageConverter();
    // 添加到converters
    converters.add(fastJsonConverter);
    converters.add(stringConverter);
}
```

https://blog.csdn.net/litte_frog

原文: https://blog.csdn.net/litte_frog/article/details/82764215