FasterXML / **jackson-databind**

Code | Issues 326 | **Pull requests 12** | Actions | Projects | Wiki | Security | Insights

findCustomEnumDeserializer by interfaces #2842

Edit | Open with

Closed **aohanhongzhi** wants to merge 1 commit into `FasterXML:master` from `aohanhongzhi:feature/enum-customer`

Conversation 10 | Commits 1 | Checks 0 | Files changed 1 | +10 −0

aohanhongzhi commented 5 days ago

[Chinese]我最近研究枚举的时候遇到一个需求就是希望在SpringBoot框架中将枚举正序列化和反序列化。希望在Controller层接收参数或者返回结果的时候，jackson对我定义的枚举正序列化和反序列化都是可以的。按照面向接口编程的思想，我对所有定义的枚举是先统一实现了一个接口，这个接口有两个方法，code()和description()。也就是枚举在序列化和反序列化的时候，我希望是按照code和description来的，不希望是按照枚举固有的name和ordinal来正反序列化和反序列化。
我阅读了jackson的源码了解到了jackson对枚举的正反序列化之后，我自定义了一个正序列化的枚举序列化器。然后注册到了jackson中，并且也是正常工作的。

```
        @Bean
        public Jackson2ObjectMapperBuilderCustomizer enumCustomizer() {
    //          将枚举转成json返回给前端
            return jacksonObjectMapperBuilder -> {
    //              自定义序列化器注入
                Map<Class<?>, JsonSerializer<?>> serializers = new LinkedHashMap<>();
                serializers.put(BaseEnum.class, new BaseEnumSerializer());
                serializers.put(Date.class, new DateJsonSerializer());
                jacksonObjectMapperBuilder.serializersByType(serializers);

    //              自定义反序列化器注入
                Map<Class<?>, JsonDeserializer<?>> deserializers = new LinkedHashMap<>();
                deserializers.put(BaseEnum.class, new BaseEnumDeserializer());
    //            deserializers.put(GenderEnum.class, new BaseEnumDeserializer());
                jacksonObjectMapperBuilder.deserializersByType(deserializers);

            };
        }
```

但是接下来我在反序列化枚举的时候，发生了一直只能由name或者ordinal来反序列化。我阅读源码后发现我自定义的反序列化器并没有生效，jackson依旧走的是默认的反序列化器，所以不是按照我预期的code或者description来反序列化的。经过阅读源代码之后发现代码中是有查找自定义序列化器的代码。一开始是先找当前类对应的序列化器，但是很明显没有找到，最后只能使用默认的反序列化器。通过源代码发现jackson并没有去查找当前类实现的接口对应的反序列化器。由于缺少这一步，我针对接口注册的反序列化器并没有被使用，因此我在jackson的源码中加入了使用接口去查找自定义序列化器的代码，用来解决这个问题。

```
                List<JavaType> interfaces = type.getInterfaces();

                for (JavaType javaType : interfaces) {
                    Class<?> rawClass = javaType.getRawClass();
                    deser = _findCustomEnumDeserializer(rawClass, config, beanDesc);
                    if (deser != null) {
                        return deser;
                    }
                }
```

这样一来我们就可以做到面向接口编程了。希望采纳。具体使用场景可以参考我的代码。
https://github.com/aohanhongzhi/SpringCloud-multiple-gradle

`findCustomEnumDeserializer by interfaces` `4500cb6`

cowtowncoder commented 5 days ago Member

Thank you for suggesting this fix.

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.
None yet

3 participants

☐ **Allow edits by maintainers**

I am not quite sure I understand what problem it would solve -- would it be possible to show a test case?
A unit test would be needed to verify that the fix works, too, and would show the problem to me.

aohanhongzhi commented 5 days ago Author

when I receive json data from http request body by SpringMVC Controller,

Controller below

```
    @PostMapping("add/body")
    public BaseResponseVO saveBody(@Valid @RequestBody UserParam userParam) {
        log.debug("\n====>当前添加的用户信息是{}", userParam);
        UserModel userModel = userService.add(userParam);
        return BaseResponseVO.success(userModel);
    }
```

the Controller need to deserialize the json data to POJO. there is a Field which is Enum type defined by myself in the POJO.

POJO below

```
@Data
public class UserParam {
    @NotBlank(message = "name不能为空")
    String name;
    @NotNull(message = "gender为1或者2")
    GenderEnum gender;
    @NotNull(message = "age不能为空")
    Integer age;
}
```

first , I need to defined a interface which would be implements by Enum.like this

```
package hxy.dream.entity.enums;

/**
 * The interface Enumerator.
 */
// 如果发现注入的bean无法解决json序列化问题，那么可以加上这个注解
//@JsonFormat(shape = JsonFormat.Shape.OBJECT)
public interface BaseEnum {
    /**
     * Code integer.
     *
     * @return the integer
     */
    Integer code();

    /**
     * Description string.
     *
     * @return the string
     */
    String description();

}
```

so that I defined a Eume name `GenderEnum.java` which implements `BaseEnum` .

```
public enum GenderEnum implements BaseEnum {
    BOY(100, "男"), GIRL(200, "女"),UNKNOWN(0, "未知");
    @EnumValue//标记数据库存的值是code
    private final Integer code;
    private final String description;

    GenderEnum(int code, String description) {
        this.code = code;
        this.description = description;
    }

    @Override
    public Integer code() {
        return code;
    }

    @Override
    public String description() {
        return description;
    }
```

maybe there are many Enum implement `BaseEnum` will be defined in project , so I hope there is a `JsonDeserializer<BaseEnum>` to deserialize the Enum which implement `BaseEnum` . not only `GenderEnum` .

actually, I found `com.fasterxml.jackson.databind.deser.BasicDeserializerFactory#createEnumDeserializer` can't find custom Enum deserializer by super interface which Enum implemented.

```
@Slf4j
@Configuration
public class BeanConfig {
    @Bean
    public Jackson2ObjectMapperBuilderCustomizer enumCustomizer() {
//        将枚举转成json返回给前端
        return jacksonObjectMapperBuilder -> {
//            自定义序列化器注入
            Map<Class<?>, JsonSerializer<?>> serializers = new LinkedHashMap<>();
            serializers.put(BaseEnum.class, new BaseEnumSerializer());
            serializers.put(Date.class, new DateJsonSerializer());
            jacksonObjectMapperBuilder.serializersByType(serializers);

//            自定义反序列化器注入
            Map<Class<?>, JsonDeserializer<?>> deserializers = new LinkedHashMap<>();
            deserializers.put(BaseEnum.class, new BaseEnumDeserializer());
//            deserializers.put(GenderEnum.class, new BaseEnumDeserializer());
            jacksonObjectMapperBuilder.deserializersByType(deserializers);

        };
    }
}
```

in current jackson code ,if I want to deserialize `GenderEnum` ,I have to register `BaseEnumDeserializer` by `GenderEnum.class` .like this

```
deserializers.put(GenderEnum.class, new BaseEnumDeserializer());
```

if there are many Enums implements `BaseEnum` ,I have to register all in `deserializers map` . I think it is so terrible
so I add code in jackson source below. I hope it can help me to find customer `BaseEnumDeserializer` by super interface not only Enum itself.

```
//            这里应该需要继续查找当前类的父接口
            List<JavaType> interfaces = type.getInterfaces();
//            再次查找自定义的序列化器
            for (JavaType javaType : interfaces) {
                Class<?> rawClass = javaType.getRawClass();
                deser = _findCustomEnumDeserializer(rawClass, config, beanDesc);
                if (deser != null) {
                    return deser;
                }
            }
```

if you have time and interesting to run my project https://github.com/aohanhongzhi/SpringCloud-multiple-gradle , it's my honor to show the feature to complete Enum deserialize by `BaseEnumDeserializer` .

cowtowncoder commented 4 days ago Member

@aohanhongzhi First of all, thank you for full explanation -- this makes sense.

Now, the challenge here is that deserializer registration relies on class(es) from Spring framework, `Jackson2ObjectMapperBuilderCustomizer` (and/or builders), and I am not sure which mechanism that uses from jackson-databind. Calls to find custom deserializers go through `com.fasterxml.jackson.databind.deser.Deserializers` implementations that are usually registered using Jackson `Module` s. Method getting called for `Enum` type would be `findEnumDeserializer()` (if nominal type is an `Enum` subtype), but to `findBeanDeserializer()` if nominal type is not Enum (or one of other "well-known" types).

So it is up to `Deserializers` implementation to find implementation: it definitely should be possible to create enum deserializers that you want, and get it registered for all subtypes. But the question is how to achieve that through Spring.

aohanhongzhi commented 3 days ago Author

@cowtowncoder In the current version, how can I implement the enumeration deserialization I want

cowtowncoder commented 2 days ago Member

@aohanhongzhi First you need to ensure your deserializer is used by Spring: I do not know exactly how that is to be done.
Once that works I can help with problems of deserializer itself, if any.

But as I said, you need to somehow register `Deserializers` implementation to `ObjectMapper` that Spring uses: it has methods:

- `findEnumDeserializer()` if declared type (in class definition, for property) is an `Enum`
- `findBeanDeserializer()` if declared types is not one of other listed types (not Enum, Collection, Map etc)

and you may need to implement both to see if type given should be handled by your deserializer.

aohanhongzhi commented 21 hours ago Author

@cowtowncoder
I debug jackson code in my project . `GenderEnum` will be go to `factory.createEnumDeserializer(ctxt, type, beanDesc);` ,because `ClassUtil.isEnumType(type.getRawClass())` is always `true` for `GenderEnum`

```
    protected JsonDeserializer<?> _createDeserializer2(DeserializationContext ctxt,
            DeserializerFactory factory, JavaType type, BeanDescription beanDesc)
        throws JsonMappingException
    {
        final DeserializationConfig config = ctxt.getConfig();
        // If not, let's see which factory method to use
        // 12-Feb-20202, tatu: Need to ensure that not only all Enum implementations get
        //    there, but also `Enum` -- latter wrt [databind#2605], polymorphic usage
//          判断反序列化的类是否为枚举
        if (ClassUtil.isEnumType(type.getRawClass())) { // type.isEnumType()) {
            logger.info("\n====>反系列化的类是枚举[{}]");
            return factory.createEnumDeserializer(ctxt, type, beanDesc);
        }
```

then. if `_findCustomEnumDeserializer(enumClass, config, beanDesc)` return null, it will be deserialize by `deser = new EnumDeserializer(constructEnumResolver(enumClass,config, beanDesc.findJsonValueAccessor()),config.isEnabled(MapperFeature.ACCEPT_CASE_INSENSITIVE_ENUMS));`

```
      // 23-Nov-2010, tatu: Custom deserializer?
        JsonDeserializer<?> deser = _findCustomEnumDeserializer(enumClass, config, beanDesc);
```

last

```
        if (deser == null) {
            deser = new EnumDeserializer(constructEnumResolver(enumClass,
                    config, beanDesc.findJsonValueAccessor()),
                    config.isEnabled(MapperFeature.ACCEPT_CASE_INSENSITIVE_ENUMS));
            logger.info("\n====>为[{}]调用系统的默认枚举反序列化器[{}]",type,deser);
        }
```

if I want to get CustomEnumDeserializer , I have to make `Enum.class` to replace original,like this:

```
        deserializers.put(Enum.class, new BaseEnumDeserializer());
        jacksonObjectMapperBuilder.deserializersByType(deserializers);
```

I think it's so crazy, I am not sure what's going to happen. I just want to deserializer Enum implemented by `BaseEnum.class` implementation which defined by myself. I think I should only register `BaseEnum.class` to deserializers to deserialize Clazz (like GenderEnum.class)which implementes BaseEnum.class

SpringBoot how to register deserializers to jackson

`com.fasterxml.jackson.databind.module.SimpleModule#addDeserializer`

```
        /**
         * Method for adding deserializer to handle specified type.
         *<p>
         * WARNING! Type matching only uses type-erased {@code Class} and should NOT
         * be used when registering serializers for generic types like
         * {@link java.util.Collection} and {@link java.util.Map}.
         */
        public <T> SimpleModule addDeserializer(Class<T> type, JsonDeserializer<? extends T> deser)
        {
            logger.info("\n====>SpringBoot注入起点[{}]===>[{}]",type,deser);
            _checkNotNull(type, "type to register deserializer for");
            _checkNotNull(deser, "deserializer");
            if (_deserializers == null) {
                _deserializers = new SimpleDeserializers();
            }
            _deserializers.addDeserializer(type, deser);
            return this;
        }
```

`com.fasterxml.jackson.databind.module.SimpleDeserializers#addDeserializer`

```
        public <T> void addDeserializer(Class<T> forClass, JsonDeserializer<? extends T> deser)
        {
            ClassKey key = new ClassKey(forClass);
            if (_classMappings == null) {
                _classMappings = new HashMap<ClassKey,JsonDeserializer<?>>();
            }
    //      =====>  全部添加到类里面了
            _classMappings.put(key, deser);
            // [Issue#227]: generic Enum deserializer?
            if (forClass == Enum.class) {
                _hasEnumDeserializer = true;
            }
        }
```

all class are registered to `_classMappings`

Last

1. I can register `Enum.class` to replace original `EnumDeserializer` , but that's not what I expected
2. it is simple to register deserializer to springboot
3. I `support to` add the below code to get `Customer Enum Deserializer` by super interface,like this

```
    //              这里应该需要继续查找当前类的父接口，然后再是查找父类
                List<JavaType> interfaces = type.getInterfaces();
    //              再次查找自定义的序列化器
                for (JavaType javaType : interfaces) {
                    Class<?> rawClass = javaType.getRawClass();
                    deser = _findCustomEnumDeserializer(rawClass, config, beanDesc);
                    if (deser != null) {
                        logger.info("\n====>依据接口[{}]找到了反序列化器[{}]", rawClass, deser);
                        return deser;
```

```
            }
        }
```

cowtowncoder commented 10 hours ago Member

@aohanhongzhi I think you are trying to dig too deep into internal handling here; regardless of whether type is `Enum` or not does not mean you could not register custom deserializer. But `SimpleModule` will probably not work since its `Deserializers` implementation (`SimpleDeserializers`) does not let you customize handling: you will need to be able to override method

```
findEnumDeserializer(...)
```

of `Deserializers` implementation registered: and in there you can use whatever logic you want -- for example one in this PR.
This logic should not be in the place PR suggests, however: I understand that it would solve your issue, but it is against the design unfortunately.

ZGAOF commented 23 minutes ago • edited

@cowtowncoder Thanks, I'll try to slove it with wrapper

```
public class SimpleDeserializersWrapper extends SimpleDeserializers {
    @Override
    public JsonDeserializer<?> findEnumDeserializer(Class<?> type, DeserializationConfig
config, BeanDescription beanDesc) throws JsonMappingException {
        JsonDeserializer<?> enumDeserializer = super.findEnumDeserializer(type, config,
beanDesc);
        if (enumDeserializer != null) {
            return enumDeserializer;
        }

        for (Class<?> typeInterface : type.getInterfaces()) {
            enumDeserializer = this._classMappings.get(new ClassKey(typeInterface));
            if (enumDeserializer != null) {
                return enumDeserializer;
            }
        }
        return null;
    }
}
```

// in spring java config

```
@Bean
public ObjectMapper objectMapper(Jackson2ObjectMapperBuilder builder) {
    SimpleDeserializersWrapper deserializers = new SimpleDeserializersWrapper();
    deserializers.addDeserializer(TypeEnum.class, new TypeEnumJsonDeserializer());

    SimpleModule simpleModule = new SimpleModule();
    simpleModule.setDeserializers(deserializers);
    simpleModule.addSerializer(TypeEnum.class, new TypeEnumJsonSerializer());

    ObjectMapper objectMapper = builder.createXmlMapper(false).build();
    objectMapper.registerModule(simpleModule);
    return objectMapper;
}
```

cowtowncoder commented 16 minutes ago Member

@ZGAOF Good work on overriding parts of `SimpleDeserializers` / `SimpleModule` -- I think this should do what you want?

aohanhongzhi commented 11 minutes ago Author

@cowtowncoder @ZGAOF thank you very much for your help , which will help me solve the problem of customizing enumeration serialization. I've been working on it for a week and I've learned a lot from it. thank you !

aohanhongzhi closed this 34 seconds ago