

收录于合集  
#Spring 7 #转换器 1



Scan to Follow

在我们日常开发工作中，经常需要进行对象的拷贝和转换。

网上虽有不少介绍对象转换的，这里结合业务场景推荐好用的工具。

01  
业务场景

我们先整理看下业务场景，从简单到复杂举例

- 1. 浅拷贝或深拷贝数据对象，用于产生一个新的对象；
- 2. 两个对象属性名和类型大部分相同，比如前端DTO转数据层PO；
- 3. 拷贝时需要忽略一些属性；
- 4. 不同类型的对象转换，比如Bean 和 Map相互转换；
- 5. 集合类型拷贝，比如两个List<DTO>之间拷贝和转换；
- 6. 两个对象之间一些属性名称和类型不同，需要自定义规则，比如字符串属性“¥ 8.8787”转换为数字 8.88，比如字符串 “8.8787%” 转换为数字 8.88

前面4种可以用 BeanUtil.copyProperties，再加点手动设置属性的代码，基本能解决。

然而第5、6种情况，比较复杂，手写代码如下图

```
java ×  TypeConversion.java ×  DyDataConverter$TemplateConvertImpl.java ×  DyGoodsRecordImportVo.java ×

DyRoomGoodsRecord dyRoomGoodsRecord = new DyRoomGoodsRecord();
dyRoomGoodsRecord.setLiveRoomNum( model.getLiveRoomNum() );
dyRoomGoodsRecord.setExplainTimes( model.getExplainTimes() );
dyRoomGoodsRecord.setFirstListingAt( model.getFirstListingAt() );
dyRoomGoodsRecord.setSalePrice( typeConversion.strToMoney( model.getSalePrice() ) );
dyRoomGoodsRecord.setTurnoverMoney( typeConversion.strToMoney( model.getTurnoverMoney() ) );
dyRoomGoodsRecord.setSoldGoodsCount( model.getSoldGoodsCount() );
dyRoomGoodsRecord.setPresellOrderCount( model.getPresellOrderCount() );
dyRoomGoodsRecord.setClickerCount( model.getClickerCount() );
dyRoomGoodsRecord.setExposeClickerRate( typeConversion.percentToRate( model.getExposeClickerRate() ) );
dyRoomGoodsRecord.setClickerBuyerRate( typeConversion.percentToRate( model.getClickerBuyerRate() ) );
dyRoomGoodsRecord.setGpmGoodsPay( typeConversion.strToMoney( model.getGpmGoodsPay() ) );
dyRoomGoodsRecord.setPreDeliveryRefundOrderCount( model.getPreDeliveryRefundOrderCount() );
if ( model.getPreDeliveryRefundMoney() != null ) {
    dyRoomGoodsRecord.setPreDeliveryRefundMoney( new BigDecimal( model.getPreDeliveryRefundMoney() ) );
}
```

自定义转换方法

这段看起来繁琐，自然是需要优化的，直观想到的是使用 BeanUtil.copyProperties + 自定义转换的几行代码，如果需要特别转换的属性有很多个呢？

这样一大段繁琐的代码，还很可能重复使用，这时候优雅的方案是 Mapstruct。

02  
Mapstruct介绍

MapStruct 是一个 Java 对象映射解决方案，可根据定义的映射规则自动生成类型安全的映射代码。它具有以下主要功能：

- 1. 自动生成对象之间的映射代码，避免手动编写大量重复的映射代码，提高映射代码的可读性和可维护性。
- 2. 支持从不同类型和命名约定之间转换，可处理大部分常见的数据类型转换，包括 Date、Enum、String 等。
- 3. 支持集合类型之间的映射，包括 List、Set 等，支持对集合元素类型进行自定义映射。
- 4. 支持对源类型和目标类型之间的字段进行自定义映射，可以使用注解或自定义方法等方式进行配置。
- 5. 支持复杂类型的嵌套映射和循环引用。
- 6. 可以与 Spring 等流行的框架集成，并支持通用的类型转换器和格式化器。

03  
Mapstruct使用

下面看看使用示范，这里针对以解决第5、6种业务场景做示范

第1步：引入 Maven 相关坐标

```
1 <dependency>
2   <groupId>org.projectlombok</groupId>
3   <artifactId>lombok</artifactId>
4   <version>1.18.10</version>
5   <scope>provided</scope>
6 </dependency>
7 <dependency>
8   <groupId>org.mapstruct</groupId>
9   <artifactId>mapstruct-jdk8</artifactId>
10  <version>1.3.0.Final</version>
11 </dependency>
12 <dependency>
13   <groupId>org.mapstruct</groupId>
14   <artifactId>mapstruct-processor</artifactId>
15   <version>1.3.0.Final</version>
16 </dependency>
```

注意：Maven插件使用3.6.0版本以上、lombok使用1.16.16版本以上，否则会出现这个错误。

```
1 No property named "xxx" exists in source parameter(s). Did you mean "id"
```

编译出错的话，确认插件版本没问题，还可以删除编译的 target目录后重新运行。

第2步：自定义属性转换方法

跟自定义工具类不同点，可使用 `@Component` 注入到Spring，`@Named("strToMoney")` 表示指定别名，在使用转换器时可通过参数 `qualifiedByName` 指定该属性转换的方法。

```
1 @Component
2 public class TypeConversion {
3     public static final String PERCENT_TO_RATE = "percentToRate";
4
5     /** 将带 ￼的字符串转化为指定精度的 BigDecimal */
6     @Named("strToMoney")
7     public BigDecimal strToMoney(String str) {
8         try {
9             BigDecimal result = new BigDecimal(str.replaceAll("¥|,", ""));
10            return result.setScale(2, RoundingMode.HALF_UP);
11        } catch (NumberFormatException e) {
12            throw new ServiceException("金额数据的格式有误，数据为: " + str);
13        }
14    }
15
16    /** 将带 %的字符串转化为指定精度的 BigDecimal */
17    @Named(PERCENT_TO_RATE)
18    public BigDecimal percentToRate(String str) {
19        try {
20            String replaceStr = str.replace("%", "");
21            return new BigDecimal(replaceStr).setScale(2, RoundingMode.HALF_UP);
22        } catch (NumberFormatException e) {
23            throw new ServiceException("百分比数据的格式有误，数据为: " + str);
24        }
25    }
26 }
```

第3步：自定义转换接口

mapper可进行字段映射，改变字段类型，指定转换的方法，包括设置默认值、日期的默认转换等。

重点是配置 `@Mapper` 和 `@Mapping` 注解，转换器默认会拷贝所有属性。

`@Mapping` 指定要特别处理的属性，`target`参数表示目标属性名，`source` 表示来源属性名(跟target相同时可省略)，`qualifiedByName` 表示使用 `TypeConversion` 类中的指定方法做转换。



```
1 public class DyDataConverter {
2     private static final TemplateConvert CONVERT = Mappers.getMapper(TemplateConvert.class);
3     /** 转换List对象 */
4     public static List<DyRoomGoodsRecord> convertFromVoList(List<DyGoodsRecordImportVo> list) {
5         return CONVERT.convert(list);
6     }
7     /** 转换单个对象 */
8     public static DyRoomGoodsRecord convertFromVo(DyGoodsRecordImportVo data) {
9         return CONVERT.convert(data);
10    }
11
12    /** 添加接口，指定使用自定义的转换类 TypeConversion */
13    @Mapper(uses = TypeConversion.class)
14    public interface TemplateConvert {
15        // 转换价格字符串
16        @Mapping(target = "salePrice", qualifiedByName = "strToMoney")
17        @Mapping(target = "turnoverMoney", qualifiedByName = "strToMoney", dateFormat = "yyyy-MM-dd HH:mm:ss")
18        @Mapping(target = "gpmGoodsPay", qualifiedByName = "strToMoney")
19        // 转换百分比为数字
20        @Mapping(target = "exposeClickerRate", qualifiedByName = TypeConversion.class)
21        @Mapping(target = "clickerBuyerRate", qualifiedByName = TypeConversion.class)
22        DyRoomGoodsRecord convert(DyGoodsRecordImportVo model);
23        List<DyRoomGoodsRecord> convert(List<DyGoodsRecordImportVo> list);
24    }
25 }
```

第4步：接着可以使用转换器了，先看看单个对象转换

```
1 DyGoodsRecordImportVo importVo = new DyGoodsRecordImportVo();
2 DyRoomGoodsRecord target = DyDataConverter.convertFromVo(importVo);
```

List对象转换如下，看起来很简洁，不需要自己写循环去转换

```
1 List<DyGoodsRecordImportVo> importList = new ArrayList<>();
2 List<DyRoomGoodsRecord> list = DyDataConverter.convertFromVoList(importList);
```

另外还支持设置默认值，将多个对象的属性转换到一个对象等。

04  
小结收尾

相信使用 lombok 的朋友会习惯，这样生成代码带来便利，编译后生成的代码在 target 目录查看，内容可参考本文的第1张示意图。

这里示范了 Mapstruct 指定转换方法，以及集合类型的转换，抛砖引玉，更多的特性待你使用发现咯。

文章若对你有帮助，谢谢你点赞、分享、在看。



技术二三

技术方案选型，日常分享二三，90后之感悟

37篇原创内容

>

公众号

收录于合集 #Spring 7

<

上一篇 · Spring事务-简洁说@Transactional的实现

People who liked this content also liked

推荐一个强大的在线画图工具

IT仔的笔记本



编译原理——词法分析

算法教程



被问懵了：MySQL 自增主键一定是连续的吗？

Java后端技术

