前言

> 说一点个人使用经验，如果有错误的地方烦请指出! 官方文档.

一、什么是**dozer**

> Dozer 是一个对象转换工具。Dozer可以在JavaBean到JavaBean之间进行递归数据复制,并且这些JavaBean可以是不同的复杂的类型

二、为什么要使用**Dozer**

> 在实际的项目系统中，我们经常存在JavaBean之间的拷贝。比如我们在DAO层，通过Do取得业务层需要的数据，将这些数据传递给 Service层的VO。Do与VO就存在典型的值拷贝。
>
> 典型的解决方案就是手动拷贝，弊端很明显，代码中充斥大量Set 和Get方法，真正的业务被埋藏值与值的拷贝之中。另一种方案就是使用BeanUtil，但BeanUtil不够很好的灵活性，又时候还不得不手动拷贝。Dozer可以灵活的对对象进行转换，且使用简单。

三、如何使用**dozer**

1、springboot 集成 dozer 的maven依赖

```
<!-- 对象映射框架 依赖包-->
<dependency>
    <groupId>com.github.dozermapper</groupId>
    <artifactId>dozer-spring-boot-starter</artifactId>
    <version>6.4.1</version>
</dependency>
```

2、springboot 的注入依赖

```
@Resource
private Mapper mapper;
```

3、dozer 的简单使用

下面是原拷贝对象

```
public class SrcUserEntity {
    private String uuid;
    private String userName;    // 用户名
    private String realName;    // 真实姓名
    private String password;    // 密码
    private String sex;         // 性别
```

```
    set()
    get()
    ......
}
```

目标拷贝对象

```
public class TarUserEntity {
    private String userName;    // 用户名
    private String realName;    // 真实姓名
    private String sex;         // 性别
    private String email;       // 邮箱
    set()
    get()
    ......
}
```

直接拷贝

```
    SrcUserEntity srcUserEntity = new SrcUserEntity();
    srcUserEntity.set()..... // 省略set值
    TarUserEntity tarUserEntity = new TarUserEntity();
    mapper.map(srcUserEntity,tarUserEntity);
```

四、进阶使用**dozer**

当原对象和目标对象属性名和类型一致时，直接转换即可，但是如果出现名称不一致，或者名称一致但是只需要转换部分字段时。如何处理？

1、xml配置基本属性映射

源拷贝对象

```
public class SrcUserEntity {
    private String uuid;
    private String srcUserName;     // 用户名
    private String srcRealName;     // 真实姓名
    private String srcPassword;     // 密码
    private String srcSex;          // 性别

    set()
    get()
```

```
    ......
}
```

目标拷贝对象

```
public class TarUserEntity {
    private String tarUserName;     // 用户名
    private String tarRealName;     // 真实姓名
    private Integer tarSex;          // 性别
    private String tarEmail;        // 邮箱
    set()
    get()
    ......
}
```

新建配置文件

```
resource
└──mappers
|   |   test-mapper.xml  //这里新建配置文件
|   application.properties
```

spring boot 设置引入配置文件

```
# 在 application.yml 中填写
dozer:
    mapping-files: classpath:mappers/**/*-mapper.xml
```

编写 test-mapper.xml 配置原对象和目标对象 的属性对应关系

```
<?xml version="1.0" encoding="UTF-8"?>

<mappings xmlns="http://dozermapper.github.io/schema/bean-mapping"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://dozermapper.github.io/schema/bean-mapping
http://dozermapper.github.io/schema/bean-mapping.xsd">

    <!-- 原对象SrcUserEntity 和 目标对象 TarUserEntity 属性对应关系  -->
    <mapping>
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TarUserEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcUserName</a>
```

```
            <b>tarUserName</b>
        </field>
        <field>
            <a>srcRealName</a>
            <b>tarRealName</b>
        </field>
        <field>
            <a>srcSex</a>
            <b>tarSex</b>
        </field>
    </mapping>
```

2、基于情景的映射

如果出现出现以下多个场景如何设置 **场景一**

> SrcUserEntity 类映射到 TarUserEntity 类时，只映射 srcUserName 到 tarUserName 其它字段均不映射

场景二

> SrcUserEntity 类映射到 TarUserEntity 类时，只映射 srcSex 到 tarSex 其它字段均不映射

如果直接配置两个场景的映射配置，启动将会报错 可以使用如下配置

```
    <mapping map-id="场景一id">
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TarUserEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcUserName</a>
            <b>tarUserName</b>
        </field>
    </mapping>

    <mapping map-id="场景二id">
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TarUserEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcSex</a>
            <b>tarSex</b>
        </field>
    </mapping>
```

使用如下方法，根据标记的`map-id`来场景化映射

```
    mapper.map(srcUserEntity,tarUserEntity,"场景一id");
```

2、单项映射

假设有以下配置

```
    <mapping map-id="场景一id">
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TarUserEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcUserName</a>
            <b>tarUserName</b>
        </field>
    </mapping>
```

一般情况情况是这么调用的

```
    # mapper.map(srcUserEntity,tarUserEntity,"场景一id");
    替换成
    mapper.map(tarUserEntity, srcUserEntity,"场景一id");
```

将源对象和目标对象 位置调换，目标对象将会映射到源对象。这是因为dozer映射默认是双向的，以下可以设置为单项映射，来避免不必要的麻烦

```
    <mapping map-id="场景一id" type="one-way">
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TarUserEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcUserName</a>
            <b>tarUserName</b>
        </field>
    </mapping>
```

3、深度属性映射

如果源对象内有嵌套对象，要把源对象内的嵌套对象内某一字段直接映射给目标对象，可以使用以下配置 **源对象**

```
public class SrcUserEntity {
    private String srcUserName;     // 用户名
    private Grade grade;            // 班级
    set()
    get()
    ......
}
```

```
// 嵌套类
public class Grade {
    private String gradeUuid;       // 班级id
    private String gradeName;       // 班级名
    set()
    get()
    ......
}
```

目标对象

```
public class TarFlatEntity {
    private String tarUserName;     // 用户名
    private String tarGradeUuid;    // 班级id
    private String tarGradeName;    // 班级名称
    set()
    get()
    ......
}
```

配置

```
    <mapping map-id="场景一id" type="one-way">
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TarFlatEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcUserName</a>
            <b>tarUserName</b>
        </field>
        <field>
            <a>grade.gradeUuid</a>
            <b>tarGradeUuid</b>
        </field>
        <field>
            <a>grade.gradeName</a>
            <b>tarGradeName</b>
        </field>
    </mapping>
```

4、集合映射

源对象

```
public class SrcList {
    private List<SrcUserEntity> srcUserEntitys
    set()
```

```
    get()
    ......
}
public class SrcUserEntity {
    private String srcUserName;     // 用户名
    private String srcRealName;     // 真实姓名
    set()
    get()
    ......
}
```

目标对象

```
public class TarList {
    private List<TarUserEntity> tarUserEntitys
    set()
    get()
    ......
}
public class TrcUserEntity {
    private String trcUserName;     // 用户名
    private String tarRealName;     // 真实姓名
    set()
    get()
    ......
}
```

配置

```
    <mapping map-id="场景一id" type="one-way">
        <class-a>com.sinosoft.entity.SrcList</class-a>
        <class-b>com.sinosoft.entity.TarList</class-b>
        <!-- map-id 填写集合内对象映射的map-id -->
        <field map-id="src-to-tar">
            <a>srcUserEntitys</a>
            <b>tarUserEntitys</b>
        </field>
    </mapping>

    <mapping map-id="src-to-tar">
        <class-a>com.sinosoft.entity.SrcUserEntity</class-a>
        <class-b>com.sinosoft.entity.TrcUserEntity</class-b>
        <!-- 用户名映射 -->
        <field>
            <a>srcUserName</a>
            <b>trcUserName</b>
        </field>
        <field>
            <a>srcRealName</a>
```

```
            <b>tarRealName</b>
        </field>
    </mapping>
```

#####5、map映射到对象 目标对象

```
public class TarUserEntity {
    private String tarUserName;     // 用户名
    private String tarRealName;     // 真实姓名
    private Integer tarSex;         // 性别
    private String tarEmail;        // 邮箱
    set()
    get()
    ......
}
```

配置

```
    <mapping map-id="场景一id" >
        <class-a>java.util.Map</class-a>
        <class-b>com.sinosoft.entity.TarUserEntity</class-b>
        <field>
            <a key="map的key">this</a>
            <b>tarUserName</b>
        </field>
    </mapping>
```

####6、更多文档请到 官方文档.