什么是DTO、VO、BO、AO、VO、Query

摘自《阿里巴巴Java开发规范》

> DO(Data Object):此对象与数据库表结构一一对应，通过 DAO 层向上传输数据源对象。
> DTO(Data Transfer Object):数据传输对象，Service 或 Manager 向外传输的对象。
> BO(Business Object):业务对象，由 Service 层输出的封装业务逻辑的对象。
> AO(ApplicationObject):应用对象，在Web层与Service层之间抽象的复用对象模型， 极为贴近展示层，复用度不高。
> VO(View Object):显示层对象，通常是 Web 向模板渲染引擎层传输的对象。
> Query:数据查询对象，各层接收上层的查询请求。注意超过 2 个参数的查询封装，禁止 使用 Map 类来传输。

为什么要使用Mapstruct

在项目中，Entity、DTO、VO、BO之间的转换，通常是两种做法:

1. 手动一个个字段的赋值
2. 使用各类BeanUtils提供的copy方法

方式1的缺点是编写代码和阅读代码都比较低效，方式2则是利用反射，性能不高，且功能太薄弱（如遇到类型不同的字段则不知道如何处理）

Mapstruct是对象复制的一个工具，使用Annotation Processing编译期间自动生成转换代码，很好的解决了上述方式的弊端。

特点如下:

- 效率高，在编译时直接生成转换代码，相当与帮你手动去一个个赋值
- 支持复杂属性赋值，场景类型会智能判断进行转换
- 支持不同字段间的赋值，通过注解实现

Mapstruct使用指南

Bean转换

默认会转换同名属性（即使类型不匹配），如果名称不同，需要使用@Mapping注解指定映射关系

```java
@Mapper(componentModel = "spring")
public interface ProductConvert {

    @Mapping(target = "productId", source = "id")
    ProductVO toVo(Product product);
}
```

引入:

```java
@Resource
private ProductConvert productConvert;

...
ProductVO productVo = productConvert.toVo(product);
```

MapStruct会在编译期间生成实现类，且类上标注@Component注解，从而能注入Spring容器

List转换

```java
@Mapper(componentModel = "spring")
public interface ProductConvert {

    List<ProductVO> toVo(List<Product> product);
}
```

特别注意：List转换方法上不能直接使用@Mapping注解，只有对象转换方法上可以定义，所以当List转换的时候包含不同名的属性时，需要先定义一个对象转换方法，再在对象转换方法上使用注解，List转换会使用对象转换方法从而生效。

```java
@Mapper(componentModel = "spring")
public interface ProductConvert {

    List<ProductVO> toVo(List<Product> product);

    @Mapping(target = "productId", source = "id")
    ProductVO toVo(Product product);
}
```