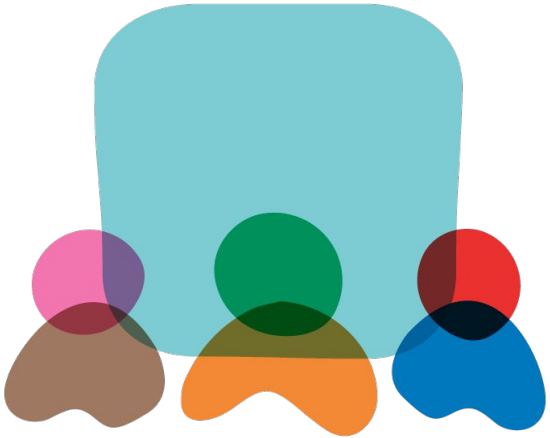
手把手教你使用对象转换工具：MapStruct

麻小遥

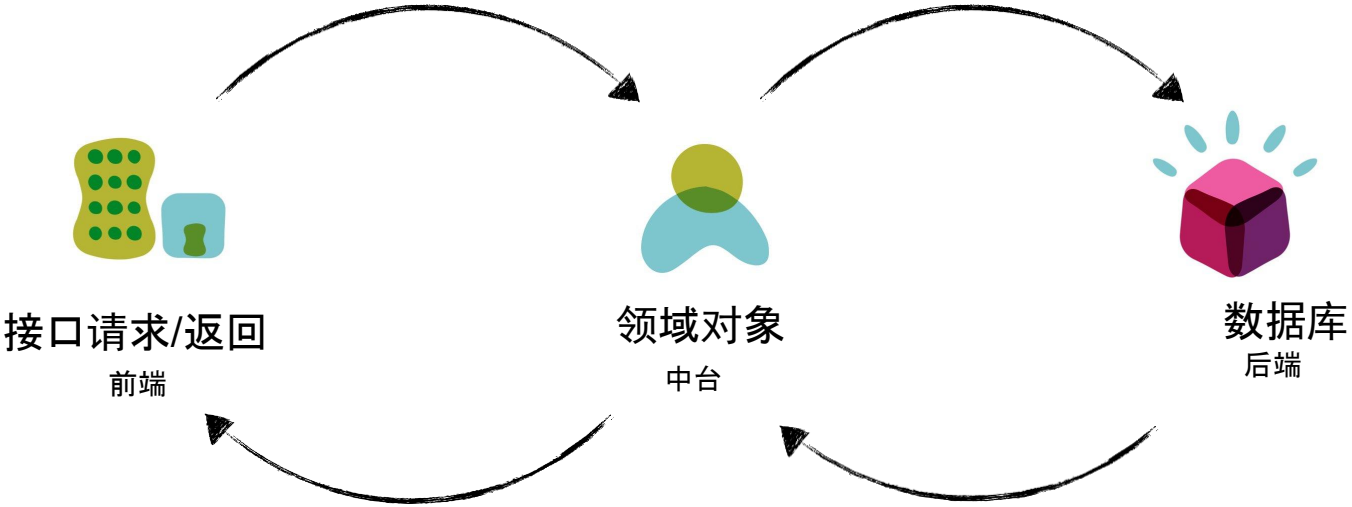
我们一起学习：

- 为何推荐使用MapStruct
- 使用MapStruct的准备工作
- 如何使用
- 优缺点讨论



你是否也有这样的痛点

痛点



痛点

```
StudentDto studentDto = new StudentDto();
studentDto.setAge(student.getAge());
studentDto.setClassroom(student.getClassroom());
studentDto.setGender(student.getGender());
studentDto.setHobbies(student.getHobbies());
studentDto.setId(student.getId());
studentDto.setName(student.getName());
studentDto.setTeacher(student.getTeacher());
```

你是如何解决的呢？

你是如何解决的:

Getter and setter

Model Mapper

Spring BeanUtils

MapStruct

哪一种性能好呢？

example1

哪一种性能好呢

Getter and setter
😊

Model Mapper
😫

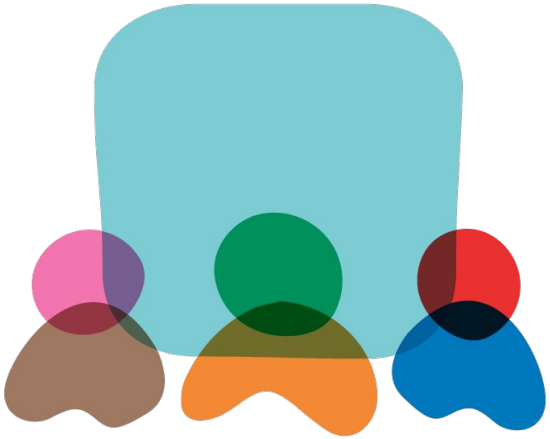
Spring BeanUtils
🙂

MapStruct
😊

教教我

教教你

- 添加依赖
- 定义接口
- 接口调用
- 编译代码
- 大功告成



教教你

- ❑ **添加依赖**
- ❑ 定义接口
- ❑ 接口调用
- ❑ 编译代码

gradle

```
plugins {
    id 'net.ltgt.apt' version '0.20'
}

apply plugin: 'net.ltgt.apt-idea'
apply plugin: 'net.ltgt.apt-eclipse'

dependencies {
    compile "org.mapstruct:mapstruct:${mapstructVersion}"
    annotationProcessor "org.mapstruct:mapstruct-processor:${mapstructVersion}"
}
```

教教你

- **添加依赖**
- 定义接口
- 接口调用
- 编译代码

maven

```
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
    <version>1.4.1.Final</version>
</dependency>


<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.1</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
        <annotationProcessorPaths>
            <path>
                <groupId>org.mapstruct</groupId>
                <artifactId>mapstruct-processor</artifactId>
                <version>1.4.1.Final</version>
            </path>
        </annotationProcessorPaths>
    </configuration>
</plugin>
```

教教你

- 添加依赖
- **定义接口**
- 接口调用
- 编译代码

使用@Mapper注解

```
package com.thoughtworks.mapstruct.example1.performance;

import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.factory.Mappers;

@Mapper
public interface CarMapper {

    CarMapper INSTANCE = Mappers.getMapper( CarMapper.class );

    @Mapping(source = "engine", target = "engineDto")
    CarDto toDto(Car car);

    EngineDto toDto(Engine engine);

}
```

教教你

- 添加依赖
- 定义接口
- **接口调用**
- 编译代码

最简单的调用

```
CarDto carDto = CarMapper.INSTANCE.toDto(car);
```

教教你

- 添加依赖
- 定义接口
- 接口调用
- **编译代码**

自动生成代码

```java
package com.thoughtworks.mapstruct.example1.performance;

import com.thoughtworks.mapstruct.example1.performance.CarDto.CarDtoBuilder;
import com.thoughtworks.mapstruct.example1.performance.EngineDto.EngineDtoBuilder;
import javax.annotation.Generated;

@Generated(
    value = "org.mapstruct.ap.MappingProcessor",
    date = "2020-11-23T21:36:57+0800",
    comments = "version: 1.4.1.Final, compiler: javac, environment: Java 1.8.0_221 (Oracle
Corporation)"
)
public class CarMapperImpl implements CarMapper {

    @Override
    public CarDto toDto(Car car) {
        if ( car == null ) {
            return null;
        }

        CarDtoBuilder carDto = CarDto.builder();

        carDto.engineDto( toDto( car.getEngine() ) );
        carDto.id( car.getId() );
        carDto.make( car.getMake() );
        carDto.numOfSeats( car.getNumOfSeats() );
        carDto.releaseDate( car.getReleaseDate() );

        return carDto.build();
    }

    @Override
    public EngineDto toDto(Engine engine) {
        if ( engine == null ) {
            return null;
        }

        EngineDtoBuilder engineDto = EngineDto.builder();

        engineDto.type( engine.getType() );

        return engineDto.build();
    }
}
```

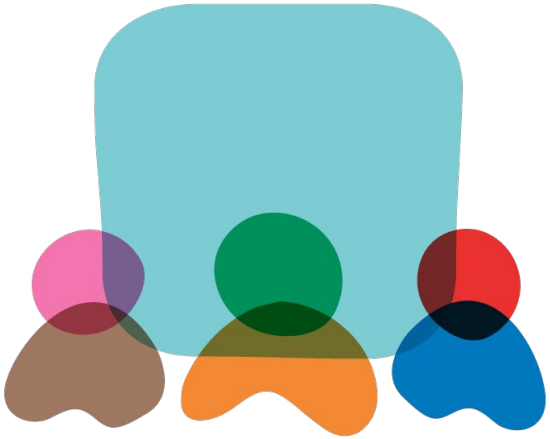
举一个简单的例子吧❗

example2

举一个简单的例子吧

- 属性名称相同
- 属性类型相同
- 无需各种自定义转换

放心大胆的无脑使用~



如果两个实体的变量名称不一致该怎么办❓

example3

如果两个实体的变量名称不一致该怎么办❓

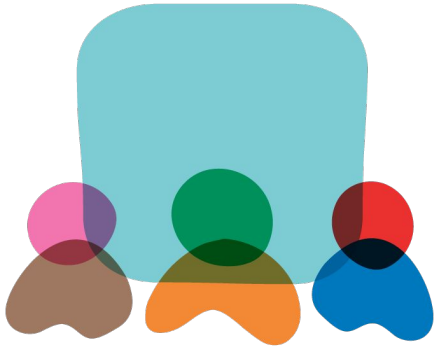
@Mapping注解来帮忙❗

属性类型不同也能转吗❓

example4

可自动转换的类型：

- Conversion between *primitive types* and their *respective wrapper types*. For example, conversion between `int` and `Integer`, `float` and `Float`, `long` and `Long`, `boolean` and `Boolean` etc.
- Conversion between *any primitive types* and *any wrapper types*. For example, between `int` and `long`, `byte` and `Integer` etc.
- Conversion between all *primitive and wrapper types* and `String`. For example, conversion between `boolean` and `String`, `Integer` and `String`, `float` and `String` etc.



不同的枚举值之间如何映射❓

example6

不同的枚举值之间如何映射❓

@ValueMapping❗

听说还能复用❓

example7

听说还能复用❓

复用:用之前写过的映射

复用

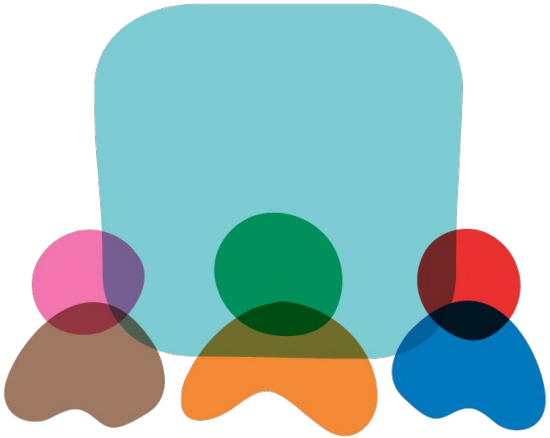
- 复用以前其他文件写过的Mapper：
  - @Mapper(uses = xxxMapper.class)
- 复用此文件写过的mapper方法：
  - @InheritInverseConfiguration

在转化过程中可以自定义转换结果吗❓

custom

自定义映射

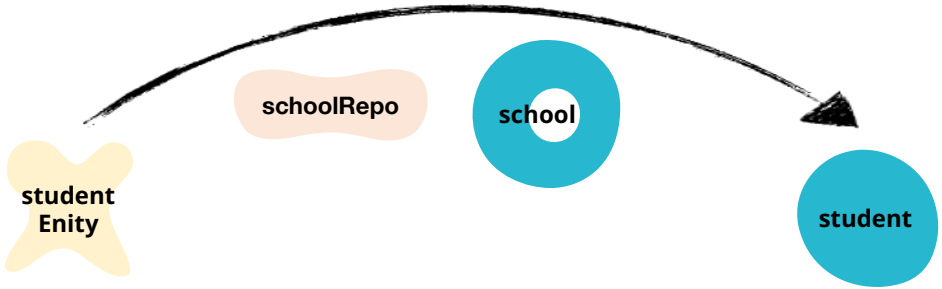
- @Named和qualifiedByName搭配使用
- expression或defaultExpression
- @BeforeMapping和@AfterMapping
- **还有一种很棒棒的方式呢～**



转化过程中，依赖其他类，需要如何实现呢❓

decorate

依赖其他类？

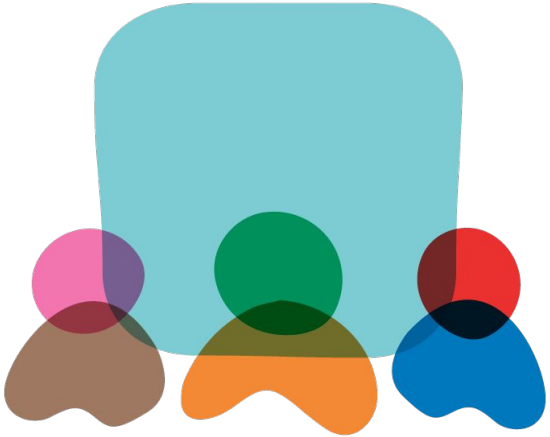


转化过程中，依赖其他类，
需要如何实现呢❓

@DecoratedWith❗

自定义映射

- @Named和qualifiedByName搭配使用
- expression或defaultExpression
- @BeforeMapping和@AfterMapping
- @DecoratedWith



异常该如何处理❓

example8

深浅拷贝❓

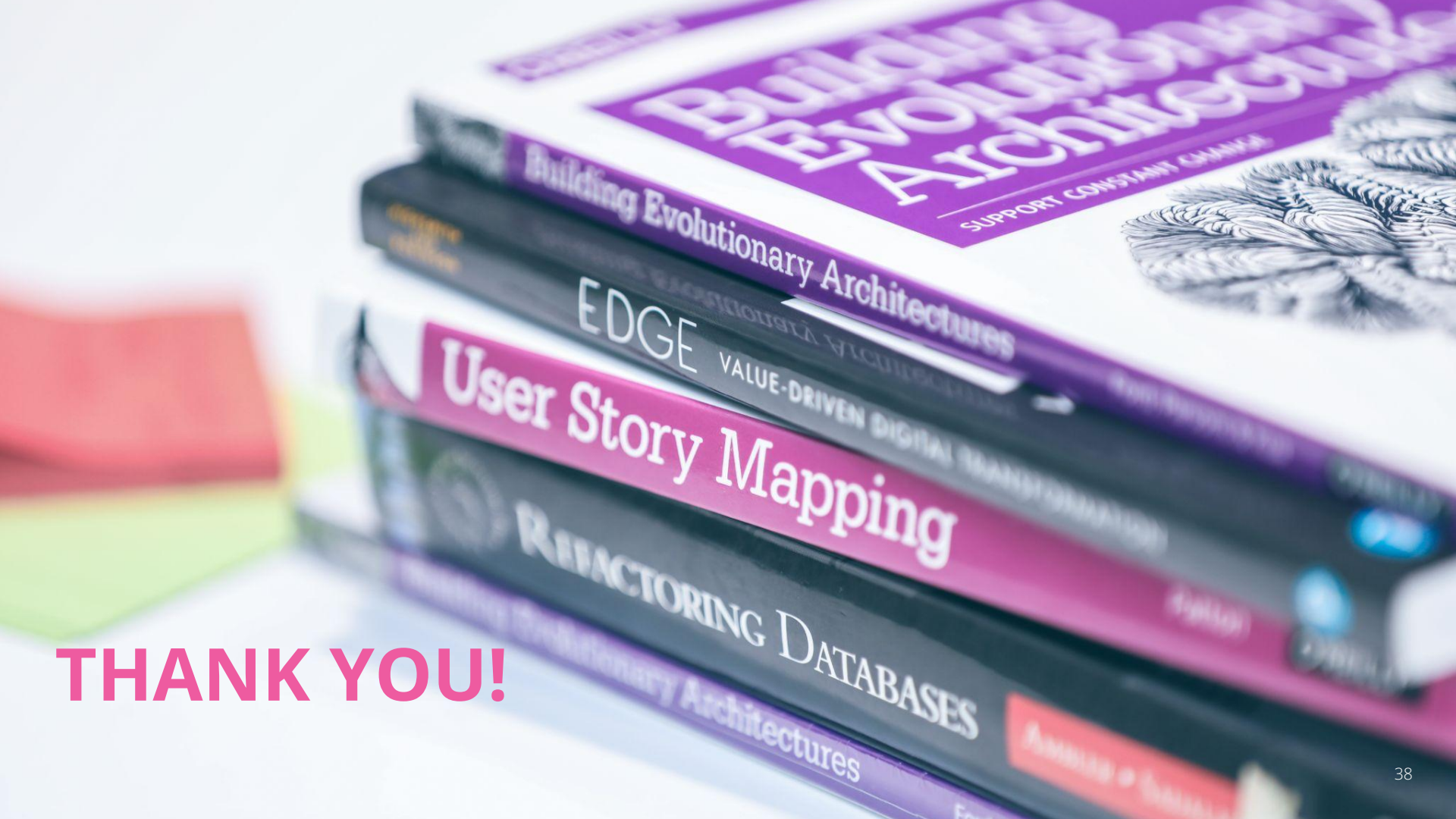
deepcopy

优缺点讨论

- 优点：
  - 性能好
  - 无需写很多get set代码
- 缺点：
  - 依赖第三方
  - 写代码时，没有对属性的校验
  - 对属性名称变化对适配性不好

参考资料:

- https://mapstruct.org/documentation/stable/reference/html/#defining-mapper
- http://www.vinsguru.com/microservices-dto-to-entity-entity-to-dto-mapping-libraries-comparison/
- https://stackabuse.com/guide-to-mapstruct-in-java-advanced-mapping-library/


Building Evolutionary Architectures
SUPPORT CONSTANT CHANGE
Building Evolutionary Architectures
EDGE
VALUE-DRIVEN DIGITAL TRANSFORMATION
User Story Mapping
REFACTORING DATABASES
Building Evolutionary Architectures

THANK YOU!