Transformer接口及其实现子类和TransformerUtils工具类

1. **Transformer<I,O>**是一个接口，存在于**org.apache.commons.collections包**中。**I表示转化器的输入类型，O表示转换器的输出类型。**

具有很多实现类：



同**Predicate接口**一样，接口定义在**org.apache.commons.collections4包中，对于其实现类，定义在了子包中。**

1. Transformer<I,O>接口只规定了一种方法，即**transform方法**。将一个输入对象input转换成另外一种对象output。



1. **自定义转换对象**：只需要实现接口Transformer<I,O>中的transform方法即可。

Transformer<Long, Date> selfTransformer = **new** Transformer<Long, Date>() {

@Override

**public** Date transform(Long milli) {

**return** **new** Date(milli);

} };

1. 实例：

**public** **static** **void** main(String[] args) {

Transformer<Long, Date> selfTransformer = **new** Transformer<Long, Date>() {

@Override

**public** Date transform(Long milli) {

**return** **new** Date(milli);

}

};

List<Long> list = **new** ArrayList<Long>();

list.add(2332334333766l);

list.add(9999999999766l);

List<Date> list2 = (List<Date>) CollectionUtils.*collect*(list, selfTransformer);

**for** (Date date : list2) {

System.***out***.println(date);

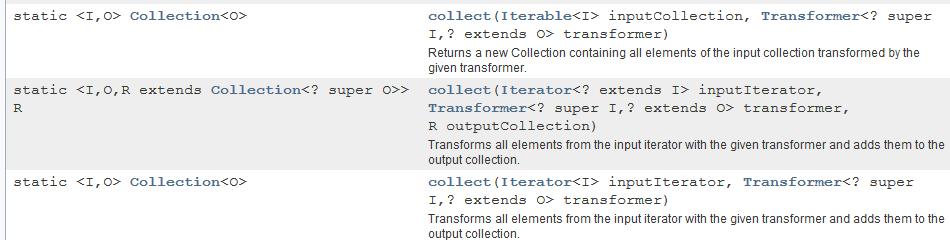
}



注意：直接使用ListUtils的transformedList方法不可以，（不知道为什么）。



使用CollectionUtils中的collect方法可以：

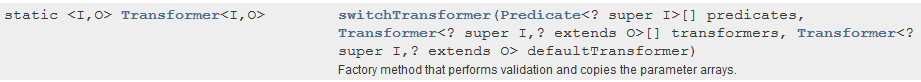


1. CollectionUtils是一个工具类，存在于org.apache.commons.collections，直接继承于java.lang.Object。
2. SwitchTransformer<I,O>类：存在于包org.apache.commons.collections4.functors包中，是Transformer接口的一个实现类。直接继承于java.lang.Object类。
3. 功能：根据Predicate对象，哪个转换器的predicate返回true，就调用那个转换器。
4. 构造方法：只有一个。

三个参数：**Predicate数组，Transformer数组，默认的转换器**。



1. **利用静态方法**获取对象：



参数同构造方法。

也可以传入一个Map集合，即Map<Predicate,Transformer>。

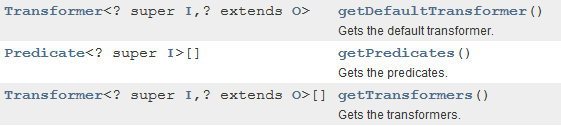


1. 成员方法：
   1. 获取三个参数的方法：

getPredicates（）：返回Predicate数组；

getTransformers（）：返回Transformer数组；

getDefaultTransformer（）：获取默认的转换器。



* 1. 执行方法: transforme()，实现Transformer接口的方法。



1. 对于SwitchTransformer进行剖析：

传入的参数是两个数组，即Predicate数组和Transformer数组，这两个数组的长度必须一样，否则报错java.lang.IllegalArgumentException: The predicate and transformer arrays must be the same size。

一个Predicate对应一个Transformer，如果Predicate返回true，则调拥对应的Transformer，否则调拥默认的转换器。

1. 示例：

**Predicate数组创建：**

Predicate<Employee> isLowSalary = **new** Predicate<Employee>() {

@Override

**public** **boolean** evaluate(Employee object) {

**return** object.getSalary()<=20000&&object .getSalary()>5000;

}

};

Predicate<Employee> isHighSalary = **new** Predicate<Employee>(){

@Override

**public** **boolean** evaluate(Employee object) {

**return** object.getSalary()>20000&& object.getSalary()<50000;

}

};

Predicate[] predicates = **new** Predicate[] {isLowSalary,isHighSalary};

Transformer数组创建：

Transformer<Employee, EmployeeLevel> lowTransformer = **new** Transformer<Employee, EmployeeLevel>() {

@Override

**public** EmployeeLevel transform(Employee input) {

**return** **new** EmployeeLevel(input.getName(), "低端初级");

}

};

Transformer<Employee, EmployeeLevel> highTransformer = **new** Transformer<Employee, EmployeeLevel>() {

@Override

**public** EmployeeLevel transform(Employee input) {

**return** **new** EmployeeLevel(input.getName(), "中高端中级");

}

};

Transformer[] transformers = **new** Transformer[] { lowTransformer,

highTransformer };

Transformer<Employee, EmployeeLevel> defaultTransformer = **new** Transformer<Employee, EmployeeLevel>() {

@Override

**public** EmployeeLevel transform(Employee input) {

**return** **new** EmployeeLevel(input.getName(), "要么高要么低，反正就是不正常");

}

};

Transformer<Employee, EmployeeLevel> switchTransformer = SwitchTransformer

.*switchTransformer*(predicates, transformers, defaultTransformer);

创建SwitchTransformer对象：

Employee employee1 = **new** Employee("马丽丽", 30000);

Employee employee2 = **new** Employee("张三丰", 10000);

Employee employee3 = **new** Employee("理无常", 60000);

Employee employee4 = **new** Employee("二百五", 2000);

List<Employee> list1 = **new** ArrayList<Employee>();

list1.add(employee1);

list1.add(employee2);

list1.add(employee3);

list1.add(employee4);

List<EmployeeLevel> list2 = (List<EmployeeLevel>) CollectionUtils

.*collect*(list1, switchTransformer);

利用迭代器迭代集合：

Iterator<EmployeeLevel> iterator = list2.iterator();

**while** (iterator.hasNext()) {

System.***out***.println(iterator.next());

}

对于Employee和EmployeeLevel就省略了。

