001/\*  
002 \* Licensed to the Apache Software Foundation (ASF) under one or more  
003 \* contributor license agreements. See the NOTICE file distributed with  
004 \* this work for additional information regarding copyright ownership.  
005 \* The ASF licenses this file to You under the Apache License, Version 2.0  
006 \* (the "License"); you may not use this file except in compliance with  
007 \* the License. You may obtain a copy of the License at  
008 \*  
009 \* http://www.apache.org/licenses/LICENSE-2.0  
010 \*  
011 \* Unless required by applicable law or agreed to in writing, software  
012 \* distributed under the License is distributed on an "AS IS" BASIS,  
013 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
014 \* See the License for the specific language governing permissions and  
015 \* limitations under the License.  
016 \*/  
017package org.apache.commons.collections4.functors;  
018  
019import java.io.Serializable;  
020  
021import org.apache.commons.collections4.Predicate;  
022import org.apache.commons.collections4.Transformer;  
023  
024/\*\*  
025 \* Transformer implementation that will call one of two closures based on whether a predicate evaluates  
026 \* as true or false.  
027 \*  
028 \* @param <I> The input type for the transformer  
029 \* @param <O> The output type for the transformer  
030 \*  
031 \* @since 4.1  
032 \*/  
033public class IfTransformer<I, O> implements Transformer<I, O>, Serializable {  
034  
035 /\*\* Serial version UID \*/  
036 private static final long serialVersionUID = 8069309411242014252L;  
037  
038 /\*\* The test \*/  
039 private final Predicate<? super I> iPredicate;  
040 /\*\* The transformer to use if true \*/  
041 private final Transformer<? super I, ? extends O> iTrueTransformer;  
042 /\*\* The transformer to use if false \*/  
043 private final Transformer<? super I, ? extends O> iFalseTransformer;  
044  
045 /\*\*  
046 \* Factory method that performs validation.  
047 \*  
048 \* @param <I> input type for the transformer  
049 \* @param <O> output type for the transformer  
050 \* @param predicate predicate to switch on  
051 \* @param trueTransformer transformer used if true  
052 \* @param falseTransformer transformer used if false  
053 \* @return the <code>if</code> transformer  
054 \* @throws NullPointerException if either argument is null  
055 \*/  
056 public static <I, O> Transformer<I, O> ifTransformer(final Predicate<? super I> predicate,  
057 final Transformer<? super I, ? extends O> trueTransformer,  
058 final Transformer<? super I, ? extends O> falseTransformer) {  
059 if (predicate == null) {  
060 throw new NullPointerException("Predicate must not be null");  
061 }  
062 if (trueTransformer == null || falseTransformer == null) {  
063 throw new NullPointerException("Transformers must not be null");  
064 }  
065  
066 return new IfTransformer<>(predicate, trueTransformer, falseTransformer);  
067 }  
068  
069 /\*\*  
070 \* Factory method that performs validation.  
071 \* <p>  
072 \* This factory creates a transformer that just returns the input object when  
073 \* the predicate is false.  
074 \*  
075 \* @param <T> input and output type for the transformer  
076 \* @param predicate predicate to switch on  
077 \* @param trueTransformer transformer used if true  
078 \* @return the <code>if</code> transformer  
079 \* @throws NullPointerException if either argument is null  
080 \*/  
081 public static <T> Transformer<T, T> ifTransformer(  
082 final Predicate<? super T> predicate,  
083 final Transformer<? super T, ? extends T> trueTransformer) {  
084  
085 if (predicate == null) {  
086 throw new NullPointerException("Predicate must not be null");  
087 }  
088 if (trueTransformer == null) {  
089 throw new NullPointerException("Transformer must not be null");  
090 }  
091  
092 return new IfTransformer<>(predicate, trueTransformer, NOPTransformer.<T>nopTransformer());  
093 }  
094  
095 /\*\*  
096 \* Constructor that performs no validation.  
097 \* Use the static factory method <code>ifTransformer</code> if you want that.  
098 \*  
099 \* @param predicate predicate to switch on, not null  
100 \* @param trueTransformer transformer used if true, not null  
101 \* @param falseTransformer transformer used if false, not null  
102 \*/  
103 public IfTransformer(final Predicate<? super I> predicate,  
104 final Transformer<? super I, ? extends O> trueTransformer,  
105 final Transformer<? super I, ? extends O> falseTransformer) {  
106  
107 super();  
108 iPredicate = predicate;  
109 iTrueTransformer = trueTransformer;  
110 iFalseTransformer = falseTransformer;  
111 }  
112  
113 /\*\*  
114 \* Transforms the input using the true or false transformer based to the result of the predicate.  
115 \*  
116 \* @param input the input object to transform  
117 \* @return the transformed result  
118 \*/  
119 @Override  
120 public O transform(final I input) {  
121 if(iPredicate.evaluate(input)){  
122 return iTrueTransformer.transform(input);  
123 }  
124 return iFalseTransformer.transform(input);  
125 }  
126  
127 /\*\*  
128 \* Gets the predicate.  
129 \*  
130 \* @return the predicate  
131 \*/  
132 public Predicate<? super I> getPredicate(){  
133 return iPredicate;  
134 }  
135  
136 /\*\*  
137 \* Gets the transformer used when true.  
138 \*  
139 \* @return the transformer  
140 \*/  
141 public Transformer<? super I, ? extends O> getTrueTransformer() {  
142 return iTrueTransformer;  
143 }  
144  
145 /\*\*  
146 \* Gets the transformer used when false.  
147 \*  
148 \* @return the transformer  
149 \*/  
150 public Transformer<? super I, ? extends O> getFalseTransformer() {  
151 return iFalseTransformer;  
152 }  
153}