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 \* Predicate implementation that transforms the given object before invoking  
026 \* another <code>Predicate</code>.  
027 \*  
028 \* @since 3.1  
029 \*/  
030public final class TransformedPredicate<T> implements PredicateDecorator<T>, Serializable {  
031  
032 /\*\* Serial version UID \*/  
033 private static final long serialVersionUID = -5596090919668315834L;  
034  
035 /\*\* The transformer to call \*/  
036 private final Transformer<? super T, ? extends T> iTransformer;  
037  
038 /\*\* The predicate to call \*/  
039 private final Predicate<? super T> iPredicate;  
040  
041 /\*\*  
042 \* Factory to create the predicate.  
043 \*  
044 \* @param <T> the type that the predicate queries  
045 \* @param transformer the transformer to call  
046 \* @param predicate the predicate to call with the result of the transform  
047 \* @return the predicate  
048 \* @throws NullPointerException if the transformer or the predicate is null  
049 \*/  
050 public static <T> Predicate<T> transformedPredicate(final Transformer<? super T, ? extends T> transformer,  
051 final Predicate<? super T> predicate) {  
052 if (transformer == null) {  
053 throw new NullPointerException("The transformer to call must not be null");  
054 }  
055 if (predicate == null) {  
056 throw new NullPointerException("The predicate to call must not be null");  
057 }  
058 return new TransformedPredicate<>(transformer, predicate);  
059 }  
060  
061 /\*\*  
062 \* Constructor that performs no validation.  
063 \* Use <code>transformedPredicate</code> if you want that.  
064 \*  
065 \* @param transformer the transformer to use  
066 \* @param predicate the predicate to decorate  
067 \*/  
068 public TransformedPredicate(final Transformer<? super T, ? extends T> transformer,  
069 final Predicate<? super T> predicate) {  
070 iTransformer = transformer;  
071 iPredicate = predicate;  
072 }  
073  
074 /\*\*  
075 \* Evaluates the predicate returning the result of the decorated predicate  
076 \* once the input has been transformed  
077 \*  
078 \* @param object the input object which will be transformed  
079 \* @return true if decorated predicate returns true  
080 \*/  
081 @Override  
082 public boolean evaluate(final T object) {  
083 final T result = iTransformer.transform(object);  
084 return iPredicate.evaluate(result);  
085 }  
086  
087 /\*\*  
088 \* Gets the predicate being decorated.  
089 \*  
090 \* @return the predicate as the only element in an array  
091 \* @since 3.1  
092 \*/  
093 @Override  
094 @SuppressWarnings("unchecked")  
095 public Predicate<? super T>[] getPredicates() {  
096 return new Predicate[] {iPredicate};  
097 }  
098  
099 /\*\*  
100 \* Gets the transformer in use.  
101 \*  
102 \* @return the transformer  
103 \*/  
104 public Transformer<? super T, ? extends T> getTransformer() {  
105 return iTransformer;  
106 }  
107  
108}