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.FunctorException;  
022import org.apache.commons.collections4.Predicate;  
023import org.apache.commons.collections4.Transformer;  
024  
025/\*\*  
026 \* Predicate implementation that returns the result of a transformer.  
027 \*  
028 \* @since 3.0  
029 \*/  
030public final class TransformerPredicate<T> implements Predicate<T>, Serializable {  
031  
032 /\*\* Serial version UID \*/  
033 private static final long serialVersionUID = -2407966402920578741L;  
034  
035 /\*\* The transformer to call \*/  
036 private final Transformer<? super T, Boolean> iTransformer;  
037  
038 /\*\*  
039 \* Factory to create the predicate.  
040 \*  
041 \* @param <T> the type that the predicate queries  
042 \* @param transformer the transformer to decorate  
043 \* @return the predicate  
044 \* @throws NullPointerException if the transformer is null  
045 \*/  
046 public static <T> Predicate<T> transformerPredicate(final Transformer<? super T, Boolean> transformer) {  
047 if (transformer == null) {  
048 throw new NullPointerException("The transformer to call must not be null");  
049 }  
050 return new TransformerPredicate<>(transformer);  
051 }  
052  
053 /\*\*  
054 \* Constructor that performs no validation.  
055 \* Use <code>transformerPredicate</code> if you want that.  
056 \*  
057 \* @param transformer the transformer to decorate  
058 \*/  
059 public TransformerPredicate(final Transformer<? super T, Boolean> transformer) {  
060 super();  
061 iTransformer = transformer;  
062 }  
063  
064 /\*\*  
065 \* Evaluates the predicate returning the result of the decorated transformer.  
066 \*  
067 \* @param object the input object  
068 \* @return true if decorated transformer returns Boolean.TRUE  
069 \* @throws FunctorException if the transformer returns an invalid type  
070 \*/  
071 @Override  
072 public boolean evaluate(final T object) {  
073 final Boolean result = iTransformer.transform(object);  
074 if (result == null) {  
075 throw new FunctorException(  
076 "Transformer must return an instanceof Boolean, it was a null object");  
077 }  
078 return result.booleanValue();  
079 }  
080  
081 /\*\*  
082 \* Gets the transformer.  
083 \*  
084 \* @return the transformer  
085 \* @since 3.1  
086 \*/  
087 public Transformer<? super T, Boolean> getTransformer() {  
088 return iTransformer;  
089 }  
090  
091}